

Micromite

Felhasználói kézikönyv

MMBasic Ver 5.4

Geoffrey R Graham

Fordította, és kiegészítette: Dr. Kónya László

For updates to this manual and more details on MMBasic
go to <http://geoffg.net/micromite.html>
and <http://mmbasic.com>

Copyright

The Micromite firmware including MMBasic and this manual are Copyright 2011-2017 by Geoff Graham.

I2C Support is Copyright 2011 Gerard Sexton.

1-Wire Support is Copyright 1999-2006 Dallas Semiconductor Corporation and 2012 Gerard Sexton.

On the Micromite Plus:

M-Stack USB/CDC driver is Copyright 2013 Alan Ott and Signal 11 Software

FatFs (SD Card) driver is Copyright (C) 2014, ChaN.

The USB VID and PIDs are sublicensed by Microchip Technology Incorporated for this project.

The compiled object code (the .hex file) for the Micromite is free software: you can use or redistribute it as you please. The source code is available via subscription (free of charge) to individuals for personal use or under a negotiated license for commercial use. In both cases go to <http://mmbasic.com> for details.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

This manual is distributed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Australia license (CC BY-NC-SA 3.0)

Contributions

Acknowledgement and thanks to Phil Boyce (WhiteWizard) for supporting the Micromite's development with much needed hardware, Peter Carnegie (G8JCF) for his help in developing the CFunction functionality and Gerard Sexton who developed the I²C and 1-Wire support for the original Maximite.

A very big thanks to Peter Mather (matherp) for his ongoing support including introducing low cost LCD panels, writing the ILI9341 and ST7735S drivers, doing the original port to the MX470 and MZ chips, porting the M Stack USB/CDC and FatFs drivers, pushing the implementation of CFunctions to new heights and the uncountable number of bugs that he has found and documented. Thank you very much Peter.

Also thanks to the members of the [Back Shed forum](#) who have beta tested the Micromite firmware over the past few years and reported many, many bugs. Thanks guys.

Tartalom

Bevezetés.....	7
Mikrovezérlő választék	8
28-lábú PIC32 mikrokontrollerek	8
44-lábú PIC32 mikrokontrollerek	8
A förmver programozása	9
28-lábú mikrokontrollerek	9
44 lábú mikrokontrollerek	9
A 28-lábú Micromite tok kivezetései	10
A 44-lábú Micromite tok kivezetései	11
Konzol kapcsolat	12
Terminál Emulátor	12
Hibakeresés	13
Gyorsbillentyűk	13
Gyors használatba vétel	14
Alapáramkör	14
Tápegység	14
Az első BASIC program	14
LED villogtatása	15
Az AUTORUN opció beállítása	15
MMBasic használata	16
Parancsok és programbevitel	16
Sorszámozás, programfelépítés és szerkesztés	16
Programok futtatása	16
Opciók beállítása	16
A Micromite speciális tulajdonságai	17
Mentett változók	17
A CPU sebességének szabályozása.....	17
CPU alvó mód	17
Watchdog időzítő.....	18
Biztonsági PIN kód.....	18
A soros konzol	19
MMBasic alapállapotba állítása.....	19
Egy önálló biztonságos HEX fájl	19
Késleltetett indulás.....	19
Grafikus LCD kijelzők	20
SPI alapú grafikus LCD Panel bekötése	20
MMBasic konfigurálása.....	21
Betölthető meghajtó programok (driverrek)	21
Érintőképernyő konfigurálása	22
Érintőképernyő kalibrálása.....	22
Érintőképernyő parancsok	22
Érintőképernyő megszakítások.....	22
Grafikus LCD panel használata	23
Csak olvasható változók.....	23

Színek.....	23
Betűkészletek (fontok)	23
Rajzoló parancsok.....	24
Példa	25
MMBasic programozása a PIC32-be	26
Kapcsolódás a konzolhoz	26
MMBasic konfigurálása	26
BASIC program betöltése	26
Speciális eszközök támogatása.....	27
Infravörös távirányító vevő dekódoló	27
Infravörös távvezérlő adó.....	28
Hőmérséklet mérése.....	28
Hőmérséklet és páratartalom mérése	29
RTC óra IC illesztése	29
Távolság mérése	30
Karakteres LCD kijelző	30
Billentyűzet illesztése	31
Kapcsolók, érintkező bemenetek	32
Elfordulás dekódoló.....	33
Szervo vezérlés	34
Teljes képernyős szövegszerkesztő	35
Színesen kódolt kijelzésű Editor.....	36
MM Edit fejlesztő környezet.....	37
MM Edit: első lépések	37
Sablonok (Fájl minta=Template) használata	38
Parancsok leírásának megjelenítése ablakban.	39
Könyvtár (Library).....	39
Auto-Backup – Automatikus mentés	39
Hibavadászat (debugging) segítése.....	39
Könnyű mozgás nagyméretű programban (könyvjelzők)	39
Fájlok méretének a csökkentése	39
Változók definiálása és használata.....	41
OPTION DEFAULT	41
OPTION EXPLICIT	41
DIM és LOCAL	42
CONST	43
Lebegőpontos és egész számok vegyítése.....	43
64-bites előjel nélküli egészek.....	43
I/O kivezetések használata.....	44
Digitális bemenetek	44
Analóg bemenetek	44
Számláló bemenetek	44
Digitális kimenetek.....	45
Impulzus szélesség moduláció (PWM).....	45
Megszakítások	45
Időzítés.....	47

Definiált szubrutinok és függvények	48
Szubrutinok paraméterei (argumentumai)	48
Helyi változók	49
Definiált függvények	49
Név szerinti paraméterátadás	50
Tömbök átadása	50
További megjegyzések	51
Példa egy definiált függvényre	51
Speciális funkciók és a könyvtár	52
Beágyazott C függvények	52
LIBRARY	52
Program inicializálás	53
MM.STARTUP	53
MM.PROMPT	53
Blokkvázlat	54
Tápellátás	55
Digitális bemenetek	55
Analóg bemenetek	55
Digitális kimenetek	55
Időzítés pontosság	55
PWM kimenet	55
Soros kommunikációs portok	55
Más kommunikációs portok	55
Flash memória újraírhatósága (endurance)	55
MMBasic jellemzők	56
Elnevezési megállapodások	56
Konstansok (állandók)	56
Műveletek és végrehajtási sorrendjük	57
Implementálási (megvalósítási) jellemzők	58
Kompatibilitás	58
Előre megadott, csak olvasható változók	59
Részletes lista	59
Parancsok	60
Részletes lista	60
Függvények	83
Részletes lista	83
Elavult parancsok és függvények	88
Részletes lista	88
Soros kommunikáció – A melléklet	89
Az OPEN parancs	89
I/O lábak kiosztása	89
Baud Rate	90
Példák	90
Olvasás és írás	90
Megszakítások	90
További soros portok	91

IEEE 485	91
Olcsó RS-232 illesztés	91
I2C kommunikáció – B melléklet.....	93
További I2C portok	95
7 és 8 bites címzés	95
10 bites címzés	95
Mester/Szolga módok	95
I/O lábak.....	95
Példa	95
Egyvezetékes kommunikáció – C melléklet.....	99
SPI kommunikáció – D melléklet	100
I/O lábak.....	100
SPI Open	100
Adatátviteli formátum	100
Szabványos Küldés/Fogadás	100
Adatfolyam Küldés/Fogadás	100
SPI Close.....	101
További SPI portok	101
Példák.....	101
SPI használata színes LCD/TFT kijelzőknél és érintőképernyő kezelésnél	101
Micromite MMBasic 5.4 Quick Reference E melléklet	103

Bevezetés



A Micromite MkII egy MMBasic förmvert tartalmazó, Microchip gyártmányú PIC32MX170 mikrovezérlő.

Az MMBasic a Microsoft BASIC Interpreter programozási nyelvvel kompatibilis megvalósítás, lebegőpontos és egész típusú számbábrázolással, karakterfüzér változókkal, tömbökkel, hosszú változónevekkel, beépített programszerkesztővel és számos további tulajdonsággal kiegészítve. Az MMBasic által támogatott kommunikációs protokollok az I²C, az egyvezetékes és az SPI busz, melyekkel sokfajta érzékelő kezelhető. Használható adatok megjelenítésére színes TFT/LCD kijelzőkön, közvetlenül képes analóg feszültségek és digitális bemenetek beolvasására, valamint a kimeneti lábain ledet és relét tud vezérelni.

A Micromite mindezt kis fogyasztás mellett biztosítja és már két darab AA méretű elemmel is működtethető. Az egyetlen szükséges külső alkatrész egy 47µF tantál vagy 10µF értékű kerámiakondenzátor. Legkönnyebben a 28-lábú DIL tokozású verziót lehet közvetlenül felhasználni, mert egy hagyományos kialakítású panel IC foglalatába helyezhető. A Micromite tulajdonságai:

- **Gyors 32 bit CPU** a hatékony MMBasic BASIC interpreter futtatására (256KiB flash programmemória és 64KiB RAM). A flash memóriában 59KiB nem felejtő memória fenntartott a program számára, és 52KiB RAM áll rendelkezésre BASIC változók, tömbök, pufferek, stb. tárolására. Ez elegendő akár 2500 soros BASIC programok számára.
- **A BASIC interpreter teljes kiépítettségű**, támogatja lebegőpontos és 64-bites egész számbábrázolású változók, karakterfüzér (string) változók, valamint a belőlük felépíthető többdimenziós tömbök, és ezekre hivatkozó hosszú változónevek használatát. A felhasználó létrehozhat saját szubrutinokat és függvényeket. A programfutás sebessége kb. 30.000 sor másodpercenként. Az MMBasic támogatja lefordított C programrészek beágyazását, amivel a végrehajtási sebesség növelhető. Egy automatikusan futtatható alkalmazás BASIC programjának a listázása és módosítása PIN-kóddal védhető.
- **Tizenkilenc I/O láb** használható a 28-lábú, és harminchárom I/O láb a 44-lábú verzióban. Ezek egymástól függetlenül konfigurálhatók akár digitális bemeneteknek vagy kimeneteknek, akár analóg bemeneteknek, akár frekvencia- vagy időmérés, akár impulzusszámlálás ellátására. Az MMBasic-ben az I/O lábak dinamikusan konfigurálhatók be- vagy kimenetek, felhúzó vagy lehúzó ellenállással, vagy anélkül. MMBasic parancsok impulzusokat hozhatnak létre, miközben ezzel párhuzamosan egyéb adatfeldolgozásra is képesek. Programmegszakításokat lehet használni bemeneti lábak állapotváltozásának a feldolgozására. Akár öt PWM vagy szervo kimenet is használható hangok előállítására, szervomotorok vezérlésére, vagy egyéb analóg vezérlési célokra.
- **TFT LCD display panelek** A TFT LCD display panelek közül az ILI9341 és a ST7735 vezérlővel szereltek támogatja, ezekkel a BASIC program számára lehetséges a szövegek kiírása, vagy vonalak, körök, dobozok rajzolása akár 65.535 színnel. Képernyőérintés érzékelésére az XPT2046 vezérlővel kiegészített rezisztív panelek használatát támogatja. Ezeknek a kisméretű érintőképernyős grafikus LCD paneleknek az ára alacsony, mégis magas színvonalú grafikus felhasználói felületet biztosítanak.
- **Programozás és vezérlés egy soros vonalon** (TTL feszültség szinteken) történik 38400 bit/sec sebességgel, ami konfigurálható. Miután a programot megírtuk, és a hibákat kijavítottuk, a Micromite úgy is beállítható, hogy bekapcsoláskor automatikusan elindítsa a BASIC programot. A programozáshoz nem szükséges speciális szoftver.
- **Teljes képernyős szövegszerkesztő** funkciót is tartalmaz a Micromite förmver. Csupán egy VT100 terminál vagy annak emulátor programja szükséges a program szerkesztéséhez és a futtatáshoz. Fejlett szövegszerkesztő funkciókat tartalmaz, mint színekkel kiemelt szintaxis, keresés, másolás, kivágás és beillesztés vágólapon keresztül.
- **A BASIC Programok könnyen másolhatók** a Micromite és Windows, Mac vagy Linux számítógépek között az XModem karakteres protokollal.
- **Számos kommunikációs protokollt támogat:** I²C, aszinkron soros, RS232, IEEE 485, SPI és az egyvezetékes. Ezek hőmérséklet, páratartalom, gyorsulás és egyéb érzékelők kiolvasására, valamint a feldolgozott adatok elküldésére használhatók.
- **Micromite parancsokkal** közvetlenül illeszthető infravörös távirányító, a DS18B20 hőmérséklet-érzékelő, LCD kijelző modulok, akkumulátorral táplált óra, numerikus billentyűzet és egyéb perifériák.
- **A mikrovezérlő speciális tulajdonságai is kihasználhatók:** Az MMBasic lehetővé teszi, hogy a CPU-t alvó állapotba küldjük, vagy átállítsuk az órajelét az energiafogyasztás és a sebesség optimalizálása miatt. A watchdog funkció ellenőrizheti a futó programot, és automatikusan újraindítja a processzort, ha programhiba történik, vagy beragad egy hurokba.
- **Tápellátás 2.3- 3.6 volt között** 6- 31 mA áramfelvétel mellett.

Mikrovezérlő választék

A szabványos Micromite förmverek a Microchip PIC32 mikrokontrollereibe programozottan működnek. Egyaránt könnyen beszerezhetők Micromite 28/44/64 modulként, vagy saját elektronikába betervezhető alkatrész-ként: www.chipcad.hu/micromite

28-lábú PIC32 mikrokontrollerek

A Micromite MkII förmverhez javasolt 28-lábú típusok:

PIC32MX170F256B-50I/SP	48MHz sebességre a gyártó által garantált típus 28-lábú DIP tokban
PIC32MX170F256B-50I/SO	48MHz sebességre a gyártó által garantált típus 28-lábú SOIC tokban
PIC32MX170F256B-50I/SS	48MHz sebességre a gyártó által garantált típus 28-lábú SSOP tokban

A legkönnyebben használható alkatrész típus a PIC32MX170F256B-50I/SP, amely DIP tokozású és garantáltan fut a maximális 48MHz-es Micromite sebességen.

A felületszerelt PIC32MX170F256B-50I/SS panelre szerelt formában is elérhető. A Micromite 28 modul a mikrokontroller mellett tartalmaz USB-soros átalakítót és 3.3V-os tápegységet, és könnyen kezelhető DIL kivezetéses formában kapható.

<https://shop.chipcad.hu/Welcome/Default.aspx?scenarioID=301&StockCode=SAJ741#TabControl-2>

A förmver korlátozottan fut PIC32MX270F256 sorozatú mikrokontrollereken is. Ezekben beépített USB interfész van, amit a Micromite nem támogat, emiatt két I/O láb (15 és 23) nem használható, továbbá a 21 és 22 lábak sem 5V toleránsak.

44-lábú PIC32 mikrokontrollerek

A Micromite MkII förmverhez javasolt 44-lábú típusok:

PIC32MX170F256D-50I/PT	48MHz sebességre a gyártó által garantált típus 44-lábú TQFP tokban
PIC32MX170F256D-50I/ML	48MHz sebességre a gyártó által garantált típus 44-lábú QFN tokban
PIC32MX170F256D-50I/TL	48MHz sebességre a gyártó által garantált típus 44-lábú VTLA tokban

A legkönnyebb a PIC32MX170F256D-50I/PT tokot használni, amely garantáltan fut 48MHz órajellel. Ez TQFP tokozású felületszerelt alkatrész 0.8 mm-es lábkiosztással. A felületszerelt alkatrész panelre szerelt formában is elérhető. A Micromite 44 modul a mikrokontroller mellett tartalmaz USB-soros átalakítót és 3.3V tápegységet könnyen kezelhető DIL kivezetéses formában:

<https://shop.chipcad.hu/Welcome/Default.aspx?scenarioID=301&StockCode=SAJ742#TabControl-2>

A Micromite MkII förmver 28 és 44-lábú mikrokontrollerekkel egyaránt használható, az egyetlen különbség, hogy a 44-lábú típusok tizennégy I/O lábbal többet biztosítanak az MMBasic programot használók számára. Itt is vannak beépített USB interfésszel rendelkező típusok, amit a Micromite nem támogat, emiatt két I/O láb nem használható.

A förmver programozása

28-lábú mikrokontrollerek

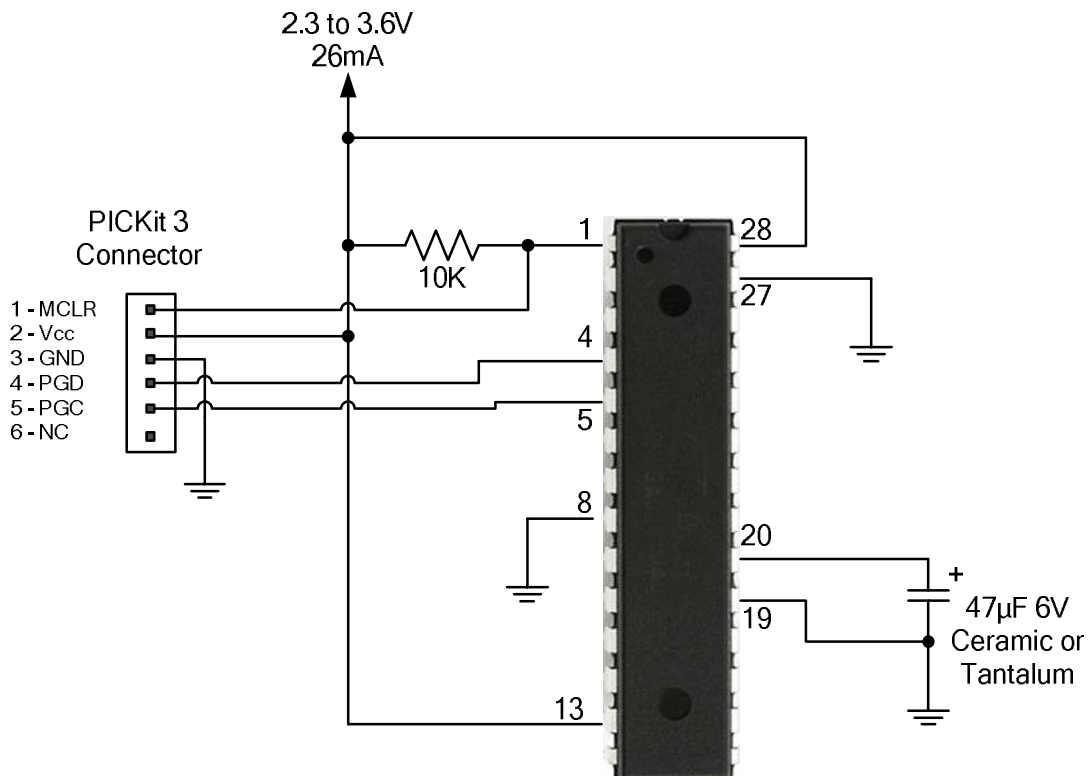
A Micromite förmver mikrokontrollerbe való programozásához megfelelő programozó szükséges. A legjobb a Microchip PICKit3 programozója. Ha telepítjük a Microchip MPLAB X (ingyenes) fejlesztő környezetét, akkor telepíthetjük az MPLAB IPE programot is, amelynek segítségével a PICKit3 már tudja programozni a tokot.

A részletek: www.geoffg.net/programming_pics.html

www.microchip.com/mplabx

<https://shop.chipcad.hu/Welcome/Default.aspx?scenarioID=301&StockCode=MIC13727#TabControl-2>

A 28-lábú tok programozásához a következő kapcsolási rajz tartozik:



A Micromite MkII és Micromite Plus förmverek programozására a Micromite 28/44/64 modulok esetén nem kell PICKit3 programozót használni, mert a modulok bemérve, a Micromite förmverekkel felprogramozottan kerülnek kereskedelmi forgalomba: www.chipcad.hu/micromite

44 lábú mikrokontrollerek

A programozás a 28-lábú változathoz hasonló. A lábak kiosztása:

PICKit 3	Leírás	44-láb
	47µF tantál vagy 10µF kerámia kondenzátor	7
1 - MCLR	Reszet (aktív nulla)	18
2 - Vcc	Tápfeszültség (3.3V)	17, 28, 40
3 - GND	Föld	6, 16, 29, 39
4- PGD	Programozás adat	21
5 - PGC	Programozás órajel	22
6 - NC	Nem használt	

Megjegyzések:

- Egy 10k felhúzó ellenállást kell kötni a MCLR láb és Vcc közé.
- Programozáskor a tok a táplálást PICKit3-ból kapja, de ajánlott egy külön táp (pl. az USB) használata.

A 28-lábú Micromite tok kivezetései

Az alábbi ábra mutatja az egyes I/O lábak lehetséges funkcióit a Micromite toknál. Megjegyezzük, hogy a fizikai lábak számozása és az MMBasic-ben használt lábszám azonos. Ez azt is jelenti, hogy kilenc láb nem használható programból, mert a tok működéséhez kellenek, amiket szürke színnel emeltünk ki:

<i>RESET Wired to +V directly or via 10K resist</i>	1	28	<i>ANALOG POWER (+2.3 to +3.6V)</i>
DIGITAL ANALOG	2	27	<i>ANALOG GROUND</i>
SPI OUT DIGITAL ANALOG	3	26	ANALOG DIGITAL PWM 2A
PWM 1A DIGITAL ANALOG	4	25	ANALOG DIGITAL SPI CLOCK
PWM 1B DIGITAL ANALOG	5	24	ANALOG DIGITAL PWM 2B
PWM 1C DIGITAL ANALOG	6	23	ANALOG DIGITAL
COM1: ENABLE DIGITAL ANALOG	7	22	DIGITAL 5V COM1: RECEIVE
<i>GROUND</i>	8	21	DIGITAL 5V COM1: TRANSMIT
COM2: TRANSMIT DIGITAL	9	20	<i>47µF TANT CAPACITOR (+)</i>
COM2: RECEIVE DIGITAL	10	19	<i>GROUND</i>
<i>CONSOLE Tx (DATA OUT)</i>	11	18	DIGITAL 5V COUNT I ² C DATA
<i>CONSOLE Rx (DATA IN)</i>	12	17	DIGITAL 5V COUNT I ² C CLOCK
<i>POWER (+2.3 to +3.6V)</i>	13	16	DIGITAL 5V COUNT WAKEUP IR
SPI IN 5V DIGITAL	14	15	DIGITAL 5V COUNT

A jelölések a következők (zárójelben a későbbiekben bemutatandó SETPIN paranccsal megadható funkciók):

ANALOG:	Bemenetek feszültség mérésére (AIN)
DIGITAL:	Digitális I/O, például digitális bemenet (DIN), digitális kimenet (DOUT) és nyitott kollektor kimenet (OOUT).
COUNT:	Bemenet frekvencia (FIN), periódusidő (PIN) mérésre vagy impulzusszámlálásra (CIN)
5V:	5V toleráns lábak. Az összes többi I/O legfőljebb 3.3V-ot visel el.
COM xxx:	Soros kommunikációra használható (ld. az A mellékletet)
I2C xxx:	I2C kommunikációra használható (ld. az b mellékletet)
SPI xxx:	SPI kommunikációra használható (ld. az d mellékletet)
PWM xxx:	PWM vagy szervo kimenetként használható
IR:	IR vevő bemenet
WAKEUP:	A CPU-t alvó üzemmódból felébresztő bemenet

A 27. és 28. lábak adják a föld és a tápfeszültség referenciát az analóg mérésekhez. Általában ezek kapcsolódnak a szokásos föld és a táp lábakhoz (8. és 13. láb), de ha zajmentes és pontos analóg mérésre van szükség, biztosítani kell, hogy a 28-as lábon jól szűrt, pontosan 3.3V tápfeszültséget kapjon. Az analóg bemeneteket a 27-es (analóg föld) lábhoz viszonyítjuk.

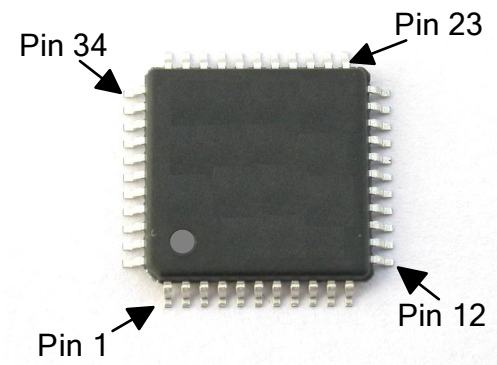
MMBasic-ben a SETPIN paranccsal lehet beállítani egy I/O láb funkcióját. A PIN parancsot vagy függvényt használjuk az adott láb kezeléséhez. Például ez a kódrészlet kiírja a 7-es láb feszültségét:

```
SETPIN 7, AIN
PRINT "A feszultseg: " PIN(7) "V"
```

Ez a feszültségmérés referenciaként a 28-as láb feszültségét használja és feltételezi, hogy a tápfeszültség ezen a lábon pontosan 3.3V. A kiolvasást skálázni kell a BASIC programban, ha más tápfeszültséget használunk.

A 44-lábú Micromite tok kivezetései

Az alábbi ábra a 44-lábú Micromite kivezetések lehetséges funkcióit mutatja minden I/O lábra. Megjegyezzük, hogy a fizikai lábak számozása, és az MMBasic-ben használt lábszám azonos. Ez azt is jelenti, hogy tizenegy láb nem használható programból, mert a tok működéséhez kellenek, amiket szürke színnel emeltük ki. A jelölés megegyezik a már leírt 28-lábú verzióval.



ANALOG DIGITAL PWM 1C	23	22	PWM 1B DIGITAL ANALOG
ANALOG DIGITAL COM1: ENABLE	24	21	PWM 1A DIGITAL ANALOG
ANALOG DIGITAL	25	20	SPI OUT (MOSI) DIGITAL ANALOG
ANALOG DIGITAL	26	19	DIGITAL ANALOG
ANALOG DIGITAL	27	18	<i>RESET Wired to +V directly or via 10K resist</i>
<i>POWER (+2.3 to +3.6V)</i>	28	17	<i>ANALOG POWER (+2.3 to +3.6V)</i>
<i>GROUND</i>	29	16	<i>ANALOG GROUND</i>
DIGITAL COM2: TRANSMIT	30	15	PWM 2A DIGITAL ANALOG
DIGITAL COM2: RECEIVE	31	14	SPI CLOCK DIGITAL ANALOG
DIGITAL 5V	32	13	5V DIGITAL
<i>CONSOLE Tx (DATA OUT)</i>	33	12	5V DIGITAL
<i>CONSOLE Rx (DATA IN)</i>	34	11	PWM 2B DIGITAL ANALOG
DIGITAL 5V	35	10	INT DIGITAL ANALOG
DIGITAL	36	9	COM1: RECEIVE 5V DIGITAL
DIGITAL 5V	37	8	COM1: TRANSMIT 5V DIGITAL
DIGITAL 5V	38	7	<i>47µF TANT CAPACITOR (+)</i>
<i>GROUND</i>	39	6	<i>GROUND</i>
<i>POWER (+2.3 to +3.6V)</i>	40	5	5V DIGITAL
DIGITAL 5V SPI IN (MISO)	41	4	5V DIGITAL
DIGITAL 5V COUNT	42	3	5V DIGITAL
DIGITAL 5V COUNT WAKEUP IR	43	2	5V DIGITAL
DIGITAL 5V COUNT I ² C CLOCK	44	1	I ² C DATA COUNT 5V DIGITAL

Konzol kapcsolat

A BASIC programok írásához, hibakereséshez és a Micromite konfigurálásához egy konzolt használunk. Ez egy kapcsolat, 38400 baud sebességgel fut és TTL szintű jeleket használ. Ez hasonló a régebbi számítógépeknél használt RS232 interfészhez, de a TTL jelszintek fordítottak, és a jelszintek 0 vagy 3.3V.

Számos USB-soros átalakító kapható. Ezek egyik oldalán soros TTL szintű jelek vannak, míg a másik oldala egy USB portra kapcsolódik. Ha ezt az átalakítót egy számítógép USB portjára csatlakoztatjuk, akkor a soros oldal egy virtuális soros portként viselkedik. Több ilyen típus közül lehet választani.

Az egyik javasolt átalakító a Microchip MCP2221 Breakout panelje:

<https://shop.chipcad.hu/Welcoming/Default.aspx?scenarioID=301&StockCode=MIC24292&ViewProduct=true&pid=1551#TabControl-2>

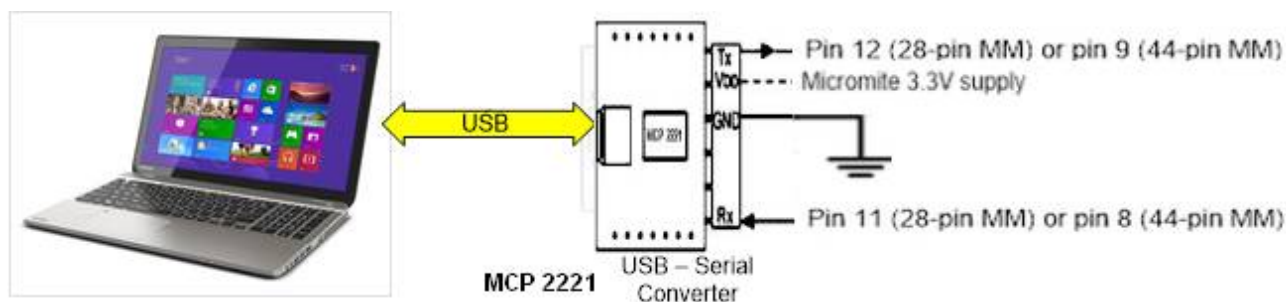
Célszerű elkerülni az FTDI FT232RL token alapuló megoldásokat, mert sok kínai gyártó lemásolta a tokot, de a jelenlegi Windows illesztőprogram ezeket nem támogatja.

A konverter soros interfész oldalán általában van egy GND (föld) kivezetés, és egy 3.3V feszültségű tápkivezetés, amelyek felhasználhatók a Micromite tápellátására. Ezekon kívül van még további jelkivezetés, a TX (vagy hasonló) jelű szolgál az adatok küldésére, és az RX jelű az adatok fogadására. A soros átalakító TX kivezetését kell a Micromite RX lábához kötni, míg az RX kivezetését a TX lábhoz.

Ha 5V-os, TTL szintű soros átalakítónk van, az is használható a Micromite-hoz. Mindössze azt kell tenni, hogy az átalakító kimeneti (TX) vonalát egy 1 kohmos ellenálláson keresztül kötjük a Micromite-hoz. Az ellenállás biztonságos szintre korlátozza az áramot.

A rajzon egy tipikus összekötés megvalósítást látjuk a MCP2221 átalakító felhasználásával:

Amikor bedugjuk az átalakító USB oldalát a számítógépbe, ekkor be kell töltenünk egy meghajtó programot hogy az átalakító működjön az operációs rendszerrel. Ha betöltjük, akkor a rendszerben megjelenik egy virtuális soros kapcsolat. Windows esetén ez megnézhető az eszközközelő (Device Manager) program segítségével ellenőrizve "Ports (COM & LPT)" menüpontot, ahol megjelenik egy új COM port.

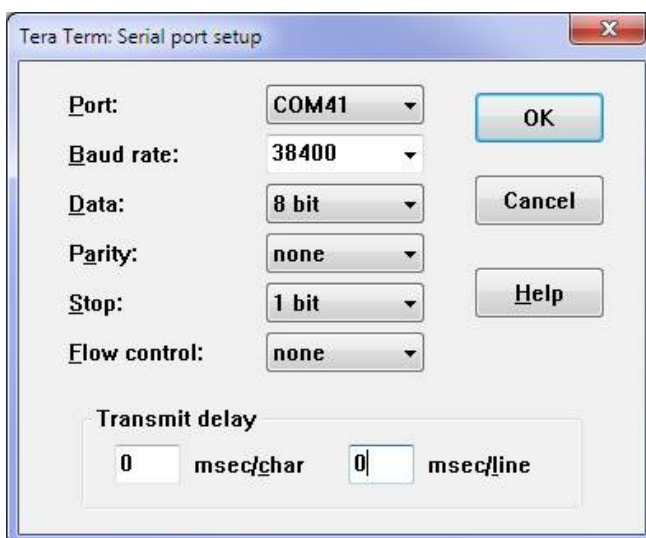


Terminál Emulátor

A számítógépünkön szükségünk van még egy terminál emulátor programra is. Ez a program úgy működik mint egy régi számítógép terminál, vagyis a távoli számítógép által küldött szöveget a képernyőn megjeleníti, és a számítógép billentyűzetén leütött karaktereket a soros vonalon elküldi a távoli számítógépnek.

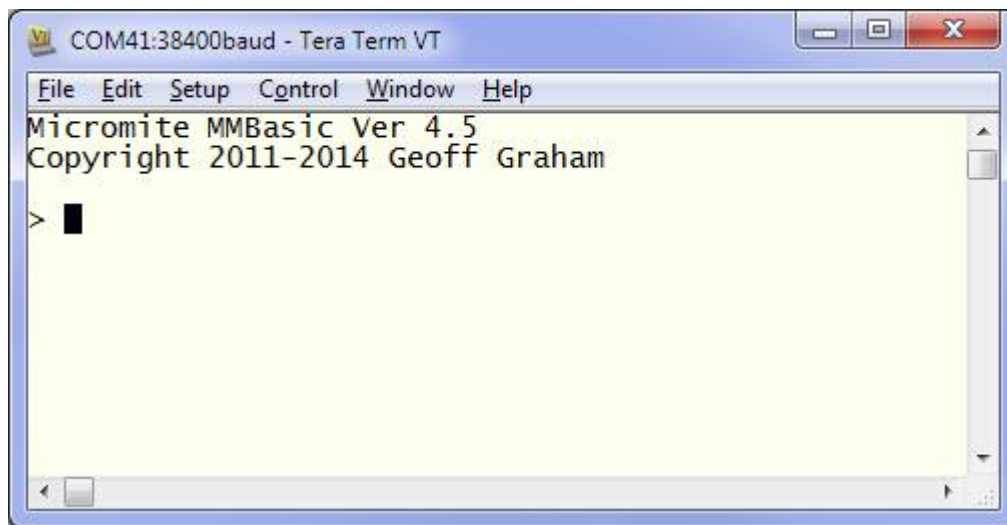
A felhasznált terminál emulátornak támogatnia kell a VT100 emulációt (működésmódot), mivel a Micromite beépített szerkesztője ezt használja. Windows esetében javasolt a Tera Term program használata, mivel ez egy jó VT100 emulátor és ismeri az XModem adatátviteli protokollt amivel át tudunk vinni mindkét irányba programokat a Micromite és a számítógép között (Tera Term letölthető: <http://tera-term.en.lo4d.com/>).

A terminál emulátor és a soros port beállításainak meg kell egyeznie a Micromite szabványos paraméterivel: 38400 baud, 8 adat bit, egy stop bit. Az ábrán látható a Tera Term beállítóablakának a mentett képe. Természetesen a "Port:" mező tartalma változhat attól függően, hogy az USB-soros átalakító meghajtó programja mit rendel hozzá a virtuális porthoz.



Ha a Tera Term programot használjuk, nem kell „Transmit delay” időt megadni az egyes karakterek átvitele között.

Ha elkészültünk a soros port és a terminál emulátor beállításával, adjuk tápot a Micromite-nak, és következőhöz hasonló kép jelenik meg a terminál emulátor ablakában:



Hibakeresés

Ha nem látjuk ezt az indítási képet, húzzuk le az USB-soros átalakító RX-TX kivezetéseit, zárjuk össze a két véget. Ha most a billentyűzeten valamit gépelünk, az a terminál ablakban meg kell jelenjen visszaekhözva. Ha ez nem történik meg akkor ez átalakító vagy a terminál emulátor hibájára utal.

Ha megjelennek a begépeltek karakterek, akkor a Micromite konzol kapcsolatában van hiba. Győződjünk meg arról, hogy a TX csatlakozik RX-hez és fordítva, és hogy az adatátviteli sebesség 38400. Ha van oszcilloszkópunk akkor megnézhetjük Micromite a TX lábán, hogy bekapcsoláskor a soros voalaon van e aktivitás, hiszen ekkor küldi el a képernyőn megjelenő karaktersorozatot.

Gyorsbillentyűk

A működést vezérlő parancsok megadása gyorsítható funkcióbillentyűkkel. Ezek a gyorsbillentyűk:

F2	RUN
F3	LIST
F4	EDIT
F10	AUTOSAVE
F11	XMODEM RECEIVE
F12	XMODEM SEND

A gomb megnyomásával a hozzá tartozó szöveg a parancssorba íródik és végrehajtódik.

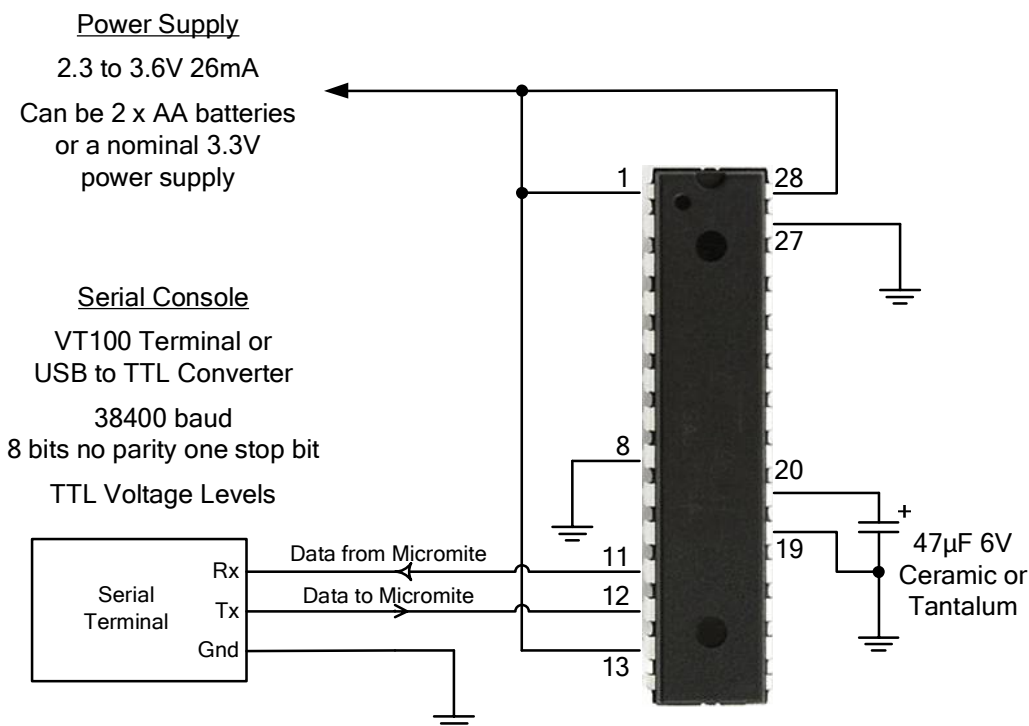
Gyors használatba vétel

A következőkben feltételezzük, hogy egy megfelelő mikrokontrollerbe beprogramoztuk a Micromite förmvert vagy kész Micromite 28 modult használunk:

<https://shop.chipcad.hu/Welcome/Default.aspx?scenarioID=301&StockCode=SAJ741#TabControl-2>

Alapáramkör

A 28-lábú Micromite alapáramköre látható az ábrán. Mivel a Micromite csak egy mikrokontrollerből és egy kondenzátorból áll, ajánlott a dugaszoló panel használata. A fejlesztés végén érdemes áttérni a végleges, nyák panel megoldásra.



Tápegység

A Micromite-hoz szükség van egy tápegységre 2.3V és 3.6V közötti kimeneti feszültséggel. Normális esetben a tápáram 26mA plusz a külső alkatrészek (ledek stb.) áramfelvétele. Két AA méretű elem vagy akkumulátor is kényelmesen használható áramforrásként, vagy használhatunk egy hagyományos tápegységet.

Általában célszerű egy 100nF-os kerámia kondenzátort közel kötni a tápegység lábához, de ez nem kritikus, és nem jelennek meg a rajzon.

A 20-as lábra kötött kondenzátor stabilizálja a PIC32 belső 1.8V-os feszültségét. Ennek jó minőségű kondenzátornak (nem elektrolit) kell lennie, és minimális értéke 10µF, amelynek az ESR soros ellenállása nem lehet 1Ω-nál nagyobb. Az ajánlott kondenzátor egy 47µF-os tantál, vagy lehet 10µF-os többretegű kerámia típus is.

A Micromite 28 modulok kiváló minőségű stabilizált tápegységgel és a szükséges járulékos alkatrészekkel is szereltek, további alkatrész nélkül azonnal használatba vehetők: www.chipcad.hu/micromite

Az első BASIC program

Feltételezve, hogy megfelelően csatlakozik a terminál emulátor a Micromite-hoz, meglátjuk a parancsokat váró jelet, ez a „nagyobb mint” szimbólum, a „>”. Vagy más néven a parancssor, amelyben megadhatunk egy tetszőleges BASIC parancsot. Majd miután megnyomjuk az Enter billentyűt, a parancs azonnal lefut.

Például, ha ez a parancs: PRINT 1/7, akkor ezt kell látni:

```
> PRINT 1/7
0.142857
>
```


Ez az úgynevezett közvetlen mód, ami rendkívül hasznos parancsok tesztelésére.

Egy BASIC program beviteléhez adjuk ki az EDIT parancsot, aminek a használatát a későbbiekben tárgyaljuk. Nézzük, jól működik-e az áramkörünk! Próbáljuk ki a következőket (a terminál emulátornak VT100-kompatibilisnak kell lennie!).

A „>” jel után gépeljük be az EDIT parancsot, amit az ENTER billentyűvel fejezzünk be.

- A szerkesztést megkezdve írjuk be a következő sort: `print "Hello Vilag"`
- Nyomjuk meg az F1 gombot a terminál emulátoron (a CTRL-Q kombináció ugyanaz). Ez jelzi a szerkesztőnek, hogy mentse a programot, és térjen vissza parancssorba.
- A parancssorba írja be RUN szöveget majd nyomjon ENTER-t.
- Ezt az üzenetet kell látnia: Hello Vilag

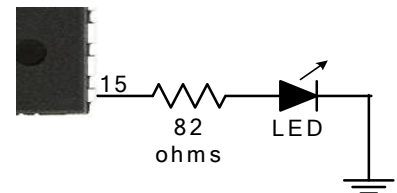
Gratulálunk! Ön most megírta és futatta az első BASIC programját a Micromite-on. Ha újra beírja az EDIT parancsot, vissza fog térni a szerkesztőbe, ahol megváltoztatható a program.

LED villogtatása

Kössünk egy LED-et a 15-ös lábra az ábrán látható módon.

Aztán az EDIT parancs kiadása után, gépeljük be a következő programot:

```
SETPIN 15, DOUT
DO
  PIN(15) = 1
  PAUSE 300
  PIN(15) = 0
  PAUSE 300
LOOP
```



Ha mentettük (F1), futtassuk ezt a programot és a LED villogni fog. Ez nem egy nagy program, de jól mutatja, hogyan tud kapcsolódni Micromite a fizikai világhoz egy BASIC programon keresztül.

A hamarosan következő "I/O láb használata" fejezetben, ezt részletesebben is bemutatjuk.

Az AUTORUN opció beállítása

Most már van egy Micromite-unk, ami valami hasznosat csinál (már ha egy villogó LED-et hasznosnak tartunk). Ha azt szeretnénk, hogy a Micromite bekapcsolásakor automatikusan elinduljon ez a program, akkor a következőket kell tennünk. A CTRL-C billentyűkombinációval megállítjuk a programot, és a parancssorba jutunk. Írjuk be a következő parancsot:

```
OPTION AUTORUN ON
```

Ez arra utasítja az MMBasic programkörnyezetet, hogy bekapcsoláskor automatikusan futtassa a programot. Teszteljük úgy, hogy megszakítjuk a táplálást és újra rákapcsoljuk. A LED villogás azonnal elkezdődik. Ha ez minden, amit szeretnénk, akkor húzzuk ki a soros konzolt és a LED villogás továbbra is működik. Ha meg akarunk változtatni a led ki-be kapcsolásának az idejét, akkor csatlakoztassuk ismét a konzolvezetéket a Micromite-hoz, a futó programot szakítsuk meg CTRL-C billentyűkombinációval, majd az EDIT parancs begépelése után a program azonnal szerkeszthető. Micromite 28 modul esetén az USB kábelt kihúzásakor mindig lépünk ki a terminál emulátor programból. Az USB kábel újbóli csatlakozása után villogni fog a led, majd program módosításához indítsuk el ismét a terminál emulátor programot.

Láthatjuk, hogy a Micromite egyik nagy előnye az, hogy nagyon könnyű rá programot írni és módosítani külön programozó készülék használata nélkül.

MMBasic használata

Parancsok és programbevitel

A parancssorban megadott parancsok azonnal végrehajtódnak. A parancssoros bevitelt arra használjuk, hogy futtassunk egy programot, vagy megadjunk egy beállítást, de ez a funkció azt is lehetővé teszi, hogy a parancsokat teszteljük.

Program bevitelére a legegyszerűbb módszer az EDIT parancs alkalmazása. Ez elindítja a Micromite-ba beépített teljes képernyős programszerkesztőt, amit a kézikönyv további részében fogunk bemutatni. Az editor fejlett képességei közé tartozik a keresés, a másolás, valamint a kivágás és a beillesztés vágólap segítségével.

A program elvileg bármilyen karakteres szövegszerkesztővel (pl. Jegyzettömb) megírható, és a szöveg a Micromite-ba vihető az XModem protokollal (ld. XMODEM parancs), a konzol soros kapcsolatán keresztül (ld. az AUTOSAVE parancs).

A harmadik és nagyon kényelmes módszer programírásra és hibakeresésre az MM Edit program használata. Ez a program Windows vagy Linux alapú számítógépen futtatható, utóbbin jelenleg csak wine emulátor segítségével. Lehetővé teszi, hogy BASIC programokat a számítógépen tárolhassunk, szerkeszthessünk, majd egyetlen egérgattintással betöltsük a Micromite memóriájába és ott ellenőrzött körülmények között futtassuk. Az MM Edit szoftvert Jim Hiley írta és a honlapjáról ingyenesen letölthető: <http://www.c-com.com.au/MMedit.htm>

Bármelyik módszerrel megírt és RAM-ba betöltött BASIC forrásprogramot elmenthetünk a Micromite belső, nem felejtő flash memóriájába. Mivel magát a BASIC forrásprogramot tárolja a flash memória, ez azt jelenti, hogy soha nem fog elveszni, még akkor sem, amikor a tápfeszültség váratlanul megszakad, vagy a processzor újraindul.

Az MMBasic korábbi változataival szemben, most már nem kell programírás közben sorszámhivatkozással ellátni a programsorokat. A parancssorba bevitt sorszámhivatkozást az MMBasic figyelmen kívül hagyja és az az ENTER gomb megnyomásakor a sorszám nélküli parancssort hajtja végre.

Sorszámozás, programfelépítés és szerkesztés

A klasszikus BASIC-nél megszokott módon a sorok elején használhatunk sorszámokat, de ez nem kötelező. A programok sorainak a felépítése a következő:

```
[line-number] [label:] command arguments [: command arguments] ...
```

A címkét, vagy sorszámot lehet használni egy adott programsor megjelölésére.

A címkével kapcsolatosan ugyanazok az előírások (hossz, karakterkészlet, stb.), mint a változók neveinél, de ez nem lehet azonos parancs-elnevezéssel.

A címke helyének a megadásakor kötelező, de a rá való hivatkozáskor már nem szükséges a kettőspont használata. Sorszámos hivatkozás esetén sem használunk kettőspontot.

Az olyan parancsoknál, mint a GOTO használhatunk címkéket, vagy sorszámot az ugrás helyének megjelölésére. Például

```
GOTO xxxx
- - -
xxxx: PRINT "Ide ugrottunk"
```

Több parancs is írható egy sorba, ezeket kettősponttal kell elválasztani egymástól. Például INPUT A: PRINT B

Programok futtatása

A program futtatása a RUN paranccsal indítható. A programot bármikor meg lehet szakítani a CTRL-C billentyűkombinációval. A program listázható a LIST paranccsal. Ez kiírja a képernyőre a programot 24 soronként felfüggesztve.

A teljes program törlése a NEW paranccsal lehetséges.

Ha bekapcsoltuk az „Autorun” funkciót (OPTION AUTORUN ON), akkor a Micromite bekapcsoláskor automatikusan elindítja és futtatja a programot.

Opciók beállítása

Az OPTION kulcsszó után számos beállítási lehetőséget adhatunk meg, ezeket jelen kézikönyv "Parancsok" fejezetében ismertetjük. Például a soros adatátviteli sebesség módosítása:

```
OPTION BAUDRATE 9600
```


A Micromite speciális tulajdonságai

Mentett változók

A Micromite nem minden esetben rendelkezik külső adattárolási rendszerrel (például SD kártya), mégis szüksége lehet arra, hogy a tápfeszültség megjelenésekor néhány változót vissza tudjon állítani. Erre a célra a VAR SAVE parancsot használhatjuk, amivel a parancsban megadott változókat adhatunk meg, hogy azokat elmentse a Micromite flash memóriájába. Az erre a célra fenntartott hely mérete 2KiB az a Micromite(MM), és 4KiB az Micromite Plus (MM+) esetén.

A mentett változók tartalma visszaállítható a VAR RESTORE parancssal, amely visszateszi az összes elmentett változót futó programunk változó táblázatába. Általában ezt a parancsot a program elején célszerű elhelyezni.

Ez alkalmas kalibrációs adatok, a felhasználó által kiválasztott opciók, beállítások és ritkán változó adatok tárolására. Intenzív használata nem javasolt, mert a flash memória elhasználódhat. A Micromite kompatibilis chipek több mint 20.000 írást és törlést tesznek lehetővé. Normál használat esetén ezt soha nem fogjuk elérni, de ha egy ciklusba tesszük, akkor gond lehet. Például egy olyan programmal, amely másodpercenként ment egy változó-csoportot, hat óra alatt tönkretesszük a flash-t. Ha programunk ugyanezt a mentést naponta egyszer végzi, akkor több mint 50 év után sem használódik el a flash memória.

Ha gyakran kell adatot menteni, akkor érdemes egy valós idejű óra (RTC) tokot használni. Ekkor az RTC SET-REG és RTC GETREG parancsokkal lehet tárolni és visszatölteni a változókat az RTC akkumulátorral védett RAM memóriáját használva. (Lásd az RTC parancsot a részletekért.)

A CPU sebességének szabályozása

Az MMBasic lehetővé teszi, hogy a Micromite órajelét a CPU parancssal megváltoztassuk. Mivel a tok áramfelvétele arányos órajelének frekvenciájával, így ez a Micromite fogyasztásának a szabályozására is lehetőséget ad. Az alapértelmezett sebesség bekapcsoláskor vagy reszet után 40MHz.

A rendelkezésre álló órajel frekvenciák és tipikus áramfelvételek:

CPU sebesség	Tipikus áramfelvétel
48 MHz	31 mA
40 MHz (alapértelmezett)	26 mA
30 MHz	21 mA
20 MHz	15 mA
10 MHz	10 mA
5 MHz	6 mA

Az MM+ sebessége akár 120 MHz lehet. (Ld. A Micromite Plus kézikönyvet a részletekért.)

Az órajel sebességének megváltoztatásakor is változatlan marad a soros portok (beleértve a konzol) adatátviteli sebessége, bár a változtatás pillanatában apró kiesés felléphet a kommunikációban.

A belső órák és időzítők frekvenciája sem változik. A PWM, SPI és I²C esetén a sebességük arányosan változni fog, de ha ezt el akarjuk kerülni, állítsuk le őket az órajelfrekvencia változtatásának idejére.

CPU alvó mód

A CPU SLEEP parancs a processzort altatja a paraméterében megadott számú másodpercig, vagy figyeli a WAKEUP láb változását, ami az ébresztésre szolgál. Alvás közben az aktuális áramfelvétel kevesebb mint 40 µA.

Normál használat esetén a CPU SLEEP *seconds* parancs kiadása után a megadott másodpercig alvó állapotba kerül a CPU. Az alvási idő után (mondjuk 5 mp), a felébredéskor, lefuttathatunk egy rövid szubrutint, és ha ez a rutin nem talál a processzornak további tennivalót, akkor azonnal ismét alvás következhet. Ilyenkor a Micromite átlagos áramfelvétele igen kicsi lesz, ami tovább csökkenthető, ha az alvás idejét megnöveljük. Másik lehetőségként az alvást megszakíthatjuk egy I/O láb állapotának a megváltoztatásával: PI. CPU SLEEP *seconds, abortpin*

A paraméter nélkül kiadott CPU SLEEP parancs automatikusan konfigurálja a WAKEUP lábat digitális bemenetnek. Alvás közben folyamatosan történik a láb vizsgálata, és a CPU felébred, amikor a bemenet állapotot vált (magas-alacsony, vagy alacsony-magas állapot váltás). Az ébresztési jel lehet egy gombnyomás, vagy más külső esemény okozta jelváltás. Az ébredés ilyenkor nagyon gyors, (<1 ms).

Az alvási funkció az IR parancssal is működik (a távirányító gombnyomásra), mely funkció megosztva használja a WAKEUP lábat a normál wakeup jellel. Ez azt jelenti, hogy egy infravörös jelet lehet használni a Micromite

ébresztésére, amely ezután dekódolja a jelet. A program ezután válaszol a távirányító gombnyomására, majd ismét alvó állapotba kerülve várja a következő távirányító parancsot.

Az áramfelvétel csökkentése SLEEP üzemmódban

Az áramot számos tényező befolyásolhatja (amelynek minimális értéke 40 μ A lehet). Ezek a következők:

- Ne hagyjuk az I/O lábakat lebegni. Ha az I/O-t lábat nem tartjuk magasan vagy alacsonyan egy külső áramkörrel, akkor állítsuk be kimenetnek.
- Húzzuk ki a konzol csatlakozást.
- Az I/O lábak alvás közben is adhatnak ki vagy nyelhetnek el áramot. SLEEP előtt győződjünk meg róla, hogy minden I/O láb olyan állapotban van, amelyik ezt nem fogja ezt tenni.
- Zárjuk be a nyitott kommunikációs csatornákat. Ez különösen a soros és az I2C esetén fontos.
- Állítsuk le a PWM vagy a SERVO kimeneteket.
- Tiltsuk le azokat az I/O lábakat, amelyeket a frekvencia, időtartam és / vagy számlálás mérésére használunk.

Ha nehézségekbe ütközünk a SLEEP alatti áramfelvétel alacsony tartásával kapcsolatban, csatlakoztassuk a Micromite-ot egy egyszerű áramkörhöz amihez semmi nem csatlakozik, és állítsuk az összes I/O lábat digitális kimenetnek. A NEW paranccsal töröljük a program memóriáját és mérjük meg a CPU SLEEP beírásakor megjelenő áramot a konzol leválasztása után. Ez a minimális áram (kb. 40 μ A-nak kell lennie), és a külső áramkörök és a szoftverek hozzáadásával mérhetjük meg az áramra gyakorolt növelő hatásukat.

Watchdog időzítő

A Micromite fő felhasználási területe a beágyazott vezérlőként való alkalmazás. A program elkészítése és tesztelése után az AUTORUN konfigurációs beállítás használható. Tápfeszültség bekapcsolásakor a tokban automatikusan elindul a program, ami valamilyen speciális feladatot végez. A felhasználónak nem kell tudni semmit arról, hogy mi fut a tok belsejében.

Ugyanakkor fennáll annak a lehetősége, hogy egy programhiba az MMBasic-ben olyan hibát eredményez, hogy az visszatér a parancssorba. Ez nem lenne jó beágyazott, folyamatosan önállóan futó programoknál, mert a Micromite már nem kapcsolódik konzolhoz. Egy másik problémát az jelenthet, hogy a program valamilyen okból végtelen ciklusba kerül. Mindkét esetben a látható hatás ugyanaz lenne, a program mindaddig állna, amíg a tápját ismét ki-be kapcsolnánk.

Ez ellen úgy védekezhetünk, hogy watchdog időzítőt használunk. Ez egy olyan időzítő, amely visszaszámol nulláig, és amikor eléri a nullát, a processzort automatikusan újraindítja (mintha a tokot ismét bekapcsolnánk). Ez megtörténik akkor is, ha az MMBasic program valamilyen okból az MMBasic parancssorban várakozik. Újraindulás után egy belső változó, az MM.WATCHDOG igaz értéket vesz föl, jelezve, hogy az újraindítás oka watchdog timeout volt.

A WATCHDOG parancsot úgy kell elhelyezni a programnak egy rendszeresen végrehajtott részében, hogy az mindig újraindítsa az időzítőt, és így az soha ne érhesse el az újraindítást okozó nulla értéket hibátlan programműködés során. Ha programhiba esetén az időzítő visszaállítása elmarad, és emiatt eléri a nullát, akkor a program újraindul (feltételezve, hogy az AUTORUN opció be van kapcsolva).

Biztonsági PIN kód

Néha fontos lehet, hogy a mikrokontrollerben tárolt bizalmas adatokat és programot megvédjük, elrejtjük. A Micromite esetében ezt az OPTION PIN xxx paranccsal érhetjük el. Ez a parancs beállít egy akár 8 számjegyből álló PIN-kódot (ami a flash memóriában tárolódik) és mielőtt a Micromite visszatérne a parancssorba (bármilyen okból) a felhasználónak a konzolon meg kell adnia a PIN-kódot. A helyes PIN megadása nélkül a felhasználó nem tud eljutni a parancssorba, és két lehetősége marad: vagy megadja a helyes PIN kódot a belépéshez, vagy újraindítja a Micromite-ot konzolba való belépés nélkül. Amikor újraindul, a felhasználótól továbbra is a helyes PIN-kód megadását kéri a parancssor eléréséhez.

Mivel a behatoló nem éri el a parancssort, nem tudja sem listázni, sem lemásolni a programot, nem tudja azt megváltoztatni, vagy bármilyen szempontból módosítani. Ha egyszer beállította a PIN kódot, a védelem kizárólag úgy szüntethető meg, hogy a helyes PIN megadásával belép a konzolra és kikapcsolja a védelmet (parancssorba beírja: OPTION PIN 0). Ha a szám elveszett, az egyetlen módszer az MMBasic resetelése, alapállapotba hozása a később leírtak szerint (amely viszont törli a programot).

Persze ha egy illetéktelen például PIC32 programozóval kiolvassa a tokot, és fáradságos, sok munkával elemezi a flash memória tartalmát, eredményre juthat. Emiatt ez nem tekinthető 100%-os biztonsági megoldásnak, de elég erős elrettentésnek tekinthető.

A soros konzol

Az OPTION BAUDRATE paranccsal a konzol adatátviteli sebességet lehet változtatni, akár maximum 230.400 bps-ig. A konzol átviteli sebességének növelése miatt a teljes képernyős szerkesztő sokkal gyorsabban tudja újrarajzolni a képernyőt. Megbízható soros kapcsolatnál érdemes a sebességet legalább 115.2 kbps sebességre növelni.

Ha megváltoztattuk a konzol átviteli sebességét az addig megmarad, amíg egy másik OPTION BAUDRATE paranccsal megváltoztatjuk. Ha rossz sebességet állítunk be, azt már a konzolon nem tudjuk átállítani, az egyetlen megoldás az MMBasic alaphelyzetbe állítása.

MMBasic alapállapotba állítása

Az MMBasic eredeti konfigurációjának visszaállítására két módszer közül választhatunk:

- A tokot újraprogramozzuk a Micromite förmverrel, egy PIC32 programozóval.
- Bekapcsoláskor egy felkiáltójel (!) sorozat küldése a konzol Rx vonalbemenetére 38400 baud sebességgel.

A Micromite bekapcsolása utáni első 100 msec során a Micromite beállítja a konzol sebességét 38400 baud-ra, és figyel, hogy érkezett-e felkiáltó jel. Ha igen, akkor vár két másodpercet, figyelve, hogy jött-e ez idő alatt legalább 30 ilyen karakter és ekkor a Micromite alaphelyzetbe állítja magát és küld egy "MMBasic reset completed" üzenetet a konzolra. Ezt egyszerűen úgy érhetjük el, a terminál emulátorban beállítjuk 38400 baudot, lenyomva tartjuk a felkiáltójel gombot, miközben bekapcsoljuk a Micromite-t. Ha a visszaállítás sikeres volt, az "MMBasic reset completed" üzenet jelenik meg a konzolon. Ez a módszer akkor is működik, ha az adatátviteli sebesség nem szabványos értékre lett állítva.

Bármelyik módszer használata esetén a programmemória és a mentett változók is teljesen törlődnek, és minden beállítás (biztonsági PIN, konzol adatátviteli sebesség, stb.) visszaáll az eredeti alapértékre.

Egy önálló biztonságos HEX fájl

Ha írunk egy programot a Micromite-ra, és kiegészítjük a következő beállításokkal,

```
OPTION BREAK 0
OPTION AUTORUN ON
```

akkor egy olyan programot kapunk, amit nem lehet megállítani vagy megszakítani. További biztonságot jelent a watchdog időzítő és PIN-kód használata.

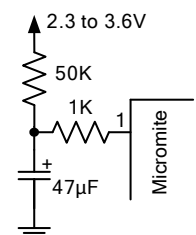
Ebben az esetben elővehetjük a PICK it3 programozót és MPLAB IPE szoftvert, hogy kiolvassa a Micromite teljes flash memória tartalmát. Ez exportálható, mint hex fájl, és tartalmazza majd a Micromite MMBasic interpretert, valamint a BASIC programot és a futtatási opciókat. Ez a fájl elküldhető bárkinek, mint egy működő alkalmazás, és ha betöltjük egy Micromite kompatibilis mikrovezérlőbe, az futni fog. Ilyenkor már megkülönböztethetetlen az, hogy ezt nem C-ben írtuk (kivéve az MMBasic által kiírt indítási szöveget). Így a felhasználó egy kompakt, működő programot kapott.

Késleltetett indulás

A Micromite mintegy 100 msec alatt indul el, de bizonyos esetekben szükséges lehet késleltetni az indulást, hogy más áramkör (például USB-soros híd) előbb induljon.

Ezt úgy lehet elérni, hogy lassítjuk az 1-es láb (28-lábú tok), vagy a 18-as láb (44-lábú tok), feszültségemelkedését, vagyis késleltetjük a Micromite reset ciklusát. Egy erre alkalmas áramkör látható a jobb oldalon. A késleltetés 50k ellenállás és 47 µF kondenzátor esetén kb. 350 msec.

Az 1k ellenállás azért van, hogy elvégezze a kondenzátor biztonságos kisütését a feszültség megszűnésekor.



Grafikus LCD kijelzők

A MicroMite támogatja az olyan színes LCD kijelző panelek használatát, amelyek ILI9341 vezérlővel és SPI interfésszel működnek. Ezeknek 240x320 pixeles színes TFT kijelzőjük van, és több méretben: 2.2", 2.4" és 2.8" elérhetők, és viszonylag olcsók. A megfelelő kijelzőket a vezérlő neve alapján érdemes keresni (ILI9341).

Tesztelt hazai típusok: <https://shop.chipcad.hu/Welcome/Default.aspx?scenarioID=360&search=tftm+rtp&search>

Az MM+ szintén támogatja ezeket a kijelzőket, de a méretük már 1.4"- 8" között lehet. Ld. a "Micromite Plus Manual" anyagot a részletekért.

Számos hasonló kijelző kapható, azonban van néhány apró, de fontos különbség, ami miatt a Micromite mégsem támogatja mindegyiket. MMBasic az ábrán látható kijelzőkkel volt tesztelve, ezért ha garantálni akarja a sikert, a fotókon láthatókat válassza, amelyek jellemzőit a későbbi táblázat tartalmazza.

Az ILI9341 alapú kijelzők SPI interfészt használnak, a következő alapvető jellemzőkkel:

- 2.2, 2.4, 2.8 vagy 3.2 colos kijelző
- Felbontás 240 x 320 pixel, színmélység 262K/65K
- A vezérlő típusa: ILI9341 SPI interfésszel.

A képen lévő kijelző érintőképernyős, amit az MMBasic teljes mértékben támogat. Ennek a kijelzőnek vannak érintést nem kezelő változatai is (a kezelő 16-lábú IC a nyomtatott áramkörtől jobbra van), de az árkülönbség a két típus között igen kicsi.

Vigyázzunk, mert néhány eBay-en forgalmazó eladó, érintésvezérlővel mutatja a panelt, de később a tulajdonság listában már az szerepel, hogy nem támogatja az érintéses kezelést.

SPI alapú grafikus LCD Panel bekötése

Az SPI buszon kommunikáló kijelző és érintésvezérlő (ha van) a Micromite SPI csatornáján osztozik a Micromite-on futó BASIC programmal. Az SPI csatorna megosztása a BASIC programot alapvetően nem befolyásolja. Az SPI kommunikáció leírását a kézikönyv D melléklete tartalmazza, ahol ezt részletesen bemutatjuk.

A következő táblázat foglalja össze az LCD kijelző és Micromite összekapcsolásához szükséges összekötéseket:

ILI9341 Kijelző XPT2046 touch szenzor	Leírás	MM28	MM44
T_IRQ	Touch megszakítás	Konfigurálható	
T_DO	Touch adat ki (MISO)	Pin 14	Pin 41
T_DIN	Touch adat be (MOSI)	Pin 3	Pin 20
T_CS	Touch kiválasztás	Konfigurálható	
T_CLK	Touch SPI órajel	Pin 25	Pin 14
SDO (MISO)	Kijelző adat ki(MISO)	Pin 14	Pin 41
LED	A háttérvilágítás táplálása (ls. lent)		
SCK	Kijelző SPI órajel	Pin 25	Pin 14
SDI (MOSI)	Kijelző adat be (MOSI)	Pin 3	Pin 20
D/C	Kijelző Adat/Parancs vezérlés	Konfigurálható	
RESET	Kijelző Reszet (aktív nulla)	Konfigurálható	
CS	Kijelző kiválasztás	Opcionális - Konfigurálható	
GND	Föld		



VCC	5V tápfeszültség (a vezérlő 10 mA áramnál kevesebbet igényel)
-----	---

Megjegyzés: Legyünk óvatosak, amikor kézbe vesszük a kijelzőt, mivel az ILI9341 vezérlő érzékeny elektrosztatikus károsításra és könnyen tönkre lehet tenni.

Ahol a Micromite kapcsolódásnál a "Konfigurálható" kifejezést szerepeltettük, ott a kiválasztott lábat az OPTION LCDPANEL vagy az OPTION TOUCH parancsban kell megadni (ld. később).

A háttérvilágítás áramellátását (a LED jelzésű kivezetés) a rendszer 5V-os tápfeszültségéről kell biztosítani, egy áramkorlátozó ellenállás közbeiktatásával. Az ellenállás tipikus értéke 18Ω amivel a LED árama kb. 63 mA lesz. Az ellenállás értéke változtatható attól függően, hogy kisebb fogyasztást, vagy nagyobb fényerőt akarunk.

Fontos: A kijelző paneleknél vegyük figyelembe, hogy az SPI porton több eszköz osztozik (kijelző, érintés vezérlő stb.). Ilyenkor minden tok kiválasztó (CS) jelet az MMBasic-ben kell konfigurálni, vagy pedig fixen le kell tiltani 3.3V-ra kötve. Ha ezt nem tesszük, akkor bármelyik „lebegő” Chip Select láb miatt az SPI buszon nem a helyes (megszólított) vezérlő fog válaszolni, hibás működést eredményezve.

MMBasic konfigurálása

A kijelző használatához az MMBasic-et be kell állítani, nem a programban, hanem a parancssorban begépett OPTION LCDPANEL parancs kiadásával.

A parancs pontos formája:

```
OPTION LCDPANEL ILI9341, orientation, D/C pin, reset pin [,CS pin]
```

Ahol:

'orientation' lehet LANDSCAPE (=fekvő), PORTRAIT (=álló), RLANDSCAPE vagy RPORTRAIT. Ezek rövidíthetők így: L, P, RL vagy RP. Az R előtag jelöli, hogy fordított, vagy „fejjel lefelé” képet mutat a kijelző.

'D/C pin' és 'reset pin' a Micromite azon I/O lábai, amit erre a célra kiválasztottunk és a kijelzőre kötöttünk. Bármelyik szabad láb használható.

'CS pin' szintén bármelyik I/O láb lehet, de ez opcionális. Ha az érintésvezérlőt nem használjuk, ez a paraméter a parancs végéről elhagyható és az LCD kijelző CS lábát fixen földre kell kötni. Ha az érintésvezérlőt használjuk, akkor ezt a lábat is meg kell adni, és a Micromite I/O lábára kell kötni.

Ezt a parancsot csak egyszer kell végrehajtani. Innentől kezdve az MMBasic automatikusan inicializálja a kijelzőt, hogy használni tudjuk induláskor vagy RESET után.

Bizonyos esetekben szükség lehet a tápfeszültség megszakítására az LCD-kijelzőnél, miközben a Micromite működik (pl. nagyobb akkumulátor működési idő érdekében), és ebben az esetben a GUI RESET LCDPANEL parancsot használhatjuk a kijelző újbóli bekapcsolására.

Ha az LCD panelt már nem akarjuk használni, akkor az OPTION LCDPANEL DISABLE paranccsal tilthatjuk le az LCD kijelzőt, és a felszabadított I/O lábakat ismét szabadon használhatjuk.

A kijelző tesztelés megtehető a GUI TEST LCDPANEL parancs kiadásával. Ilyenkor láthatunk gyorsan mozgó, egymásra rajzolt, színes köröket. A konzol betűköz billentyűjének megnyomásakor a teszt véget ér.

Fontos: Ez a teszt nem működik, ha a kijelzőnek van érintésvezérlője, de ezt nem konfiguráltuk. (pl. a CS lába lebeg). Ilyen esetben konfiguráljuk az érintővezérlőt az előbbieken leírtak szerint és ismét gépeljük be: GUI TEST LCDPANEL.

Megjegyzés: CPU órajelének minimum 20 MHz-nek vagy nagyobbaknak kell lenni.

A beállított konfiguráció ellenőrzéséhez használhatjuk az OPTION LIST parancsot, ami minden beállítást listáz, és így természetesen az LCD kijelző beállításait is.

Betölthető meghajtó programok (driverok)

Betölthető meghajtó programokat használhatunk olyan LCD kijelzők esetén, amelyek az ajánlottól különböző típusú kijelző vezérlőket használnak. Ezek a driverek az MMBasic förmver disztribúciót tartalmazó tömörített zip fájl CFunctions alkönyvtárában található. A csatolt dokumentáció részletesen ismerteti a driver használatba vételét, és a kivezetések bekötését.

Érintőképernyő támogatása

A legtöbb ILI9341 alapú LCD panelen van egy hozzá illesztett, rezisztív érintőképernyő, annak vezérlőjével együtt. Az MMBasic érintésérzékelésének használatához először ezt is a Micromite-al kell összekötni az előbbieken leírtak szerint, és a következő módon konfigurálni:

Érintőképernyő konfigurálása

Az MMBasic érintéskezelésének használatához, parancssorban kell kiadni az OPTION TOUCH konfiguráló parancsot (nem a programban!). Ezt azután kell végrehajtani, ha már az LCD panelt az előbbieken leírtak szerint konfiguráltuk.

A parancs:

```
OPTION TOUCH T_CS pin, T_IRQ pin
```

Ahol: 'T_CS pin' és 'T_IRQ pin' lábak a Micromite I/O lábai, amit a vezérlő kiválasztásához és az érintés megszakítás-érzékeléséhez használunk. (Bármelyik szabad láb használható).

Ezt a parancsot csak egyszer kell végrehajtani, mert a működési paramétereket a flash memória tárolja. Minden alkalommal, amikor a Micromite újraindul, az MMBasic automatikusan inicializálja az érintésvezérlőt, hogy használni tudjuk.

Ha az érintéskezelést már nem akarjuk használni, akkor az OPTION TOUCH DISABLE parancssal tilthatjuk le az érintésvezérlőt, és a felszabadított I/O lábakat ismét szabadon használhatjuk (a T_CS lábat a panelen, fixen 3.3V-ra kell kötni, hogy a vezérlő működését letiltsuk.)

Érintőképernyő kalibrálása

Mielőtt használnánk az érintési lehetőségeket, előtte be kell állítani az aktív területet a GUI CALIBRATE parancssal. A parancs megjelenít egy célmezőt a képernyő bal felső sarkában. Használjunk egy tollszerű mutató eszközt, ami nem túl hegyes, pl. műanyagból van, bökjünk vele finoman a cél közepére, és tartsuk ott a tollat lenyomva legalább egy másodpercig. Az MMBasic eltárolja ezt a helyet. Folytassuk ezt a kalibrálást a képernyő másik három sarkán, úgy ahogy a cél körbevándorol. Ha a kalibráló rutin figyelmeztethet, hogy a kalibráció nem volt pontos, ettől függetlenül használhatjuk az érintőképernyőt, de jobb, ha megismételjük a beállítási folyamatot egy kissé gondosabban.

Kalibráció után tesztelhetjük a beállítást a GUI TEST TOUCH parancssal. A parancs letörli a képernyőt, sötét lesz, és egy érintésre vár. Ha ez megtörténik, akkor egy fehér pont jelenik meg a kijelzőn, az általa érzékelt pozícióban. Ha jól kalibráltunk, akkor a pont helye pontosan meg fog egyezni azzal a hellyel, ahol a tollal a képernyőt megérintettük. A tesztből úgy tudunk kilépni, ha konzolon a betűköz billentyűt megnyomjuk.

Érintőképernyő parancsok

Az érintőképernyőhöz, programunkban a következő függvényeket használhatjuk:

- ☐ TOUCH(X)
Az éppen megérintett hely X koordinátájával tér vissza.
- ☐ TOUCH(Y)
Az éppen megérintett hely Y koordinátájával tér vissza.

Mindkét függvény "-1" értéket ad vissza, ha a képernyőt nem érintettük meg. A Micromite Plus számos további parancsot biztosít. . Ld. a "Micromite Plus Manual" anyagot a részletekért.

Érintőképernyő megszakítások

A megszakítást egy olyan IRQ lábhoz rendelhetjük, amit akkor adtunk meg, amikor az érintőképernyőt konfiguráltuk. Azért, hogy egy érintést érzékeljünk, a megszakítást INTL (azaz, magas-alacsony változás) érzékelésére kell beállítani. Például, ha konfiguráláskor az OPTION TOUCH 7, 15 beállításokat használtuk, akkor a következő program kiírja bármelyik képernyőérintés X és Y koordinátáit a konzol képernyőjére:

```
SETPIN 15, INTL, MyInt  
DO : LOOP  
SUB MyInt  
    PRINT TOUCH (X) TOUCH (Y)  
END SUB
```

A beállított megszakítás törölhető a SETPIN láb, OFF parancssal.

Grafikus LCD panel használata

Összesen nyolc rajzoló parancs áll rendelkezésre, hogy az MMBasic programunkból vezérelhessük a hozzákapcsolt LCD kijelzőt. A Micromite Plus szintén támogatja ezeket a parancsokat, és még számos további GUI parancsot. Ld. a "Micromite Plus Manual" anyagot a részletekért.

A képernyőn minden koordináta, és a méretek megadása is pixeleken történik, ahol az X koordináta a horizontális (vízszintes), az Y koordináta a vertikális (függőleges) pozíció. A bal felső sarokban van a képernyő X=0 és Y=0 pontja, és az értékek lefelé, illetve jobbra növekszenek.

Csak olvasható változók

Összesen négy, csak olvasható változó van, ami a csatlakoztatott képernyőre vonatkozó fontos információkat tartalmazza:

- MM.HRES
A képernyő szélességét (X tengely) tartalmazza pixeleken.
- MM.VRES
A képernyő magasságát (Y tengely) tartalmazza pixeleken.
- MM.FONTHEIGHT
Visszaadja az aktuális alapértelmezés szerinti font magasságát (pixeleken). A betűkészlet minden karakterének azonos a magassága.
- MM.FONTWIDTH
Visszaadja az aktuális alapértelmezés szerinti font szélességét (pixeleken). A betűkészlet minden karakterének azonos a szélessége.

Színek

A szín értéke egy 24 bites szám (true colour), ahol felső nyolc bit értéke adja a vörös szín erősségét, a középső nyolc bit a zöld szín intenzitását, míg a legalsó nyolc bit hordozza a kék szín erősségét. A legkönnyebb mód egy ilyen szám generálására az RGB() függvény, aminek formája:

```
RGB(red, green, blue)
```

A színmegadásnál a nulla érték jelöli a feketét, és 255 a szín teljes erősségét. Az RGB() függvényben a színt egyszerűsítve is megjelölhetjük, a kívánt szín angol nevét megadva, például, RGB(red) vagy RGB(cyan). Az egyszerűsített megadáshoz használható színek: white, black, blue, green, cyan, red, magenta, yellow, brown és gray. Az MMBasic automatikusan átalakít minden színt arra a formátumra, amelyet a kijelző vezérlője igényel. Ez például az ILI9341 vezérlő esetén 65535 szín 565 formátumban.

Megjegyezzük, hogy a szín (azaz az RGB() függvény által hordozott 24 bites szám túl nagy ahhoz, hogy pontosan tárolható legyen egy lebegőpontos változóban, helyette hanem a változót integer típusnak kell deklarálni.. Hasonlóképpen, az argumentumokat is egész számként kell megadni, ha egy színértéket átadunk az alprogramnak vagy a függvénynek.

A COLOUR paranccsal beállítható az alapértelmezés azoknál a parancsoknál, amelyek színparamétereket igényelnek. Ez akkor jó, ha a programunk következetes színsémákat használ, amit beállíthatunk alapértelmezettre, és akkor a rajzoló parancsok rövidített verzióját használhatjuk mindenhol a programban.

A COLOUR parancs formátuma:

```
COLOUR foreground-colour, background-colour 'előter színe, háttér színe
```

Betűkészletek (fontok)

Az MMBasic része egy beépített betűkészletet, mely 8 pixel széles és 13 pixel magas betűkből áll, magába foglalja mind a 95 szabványos (ékezet nélküli) ASCII karaktert: a 60H (decimális 96) értékű karaktert a fok szimbólummal (°) helyettesíti. Ere a betűkészletre font #1-ként hivatkozunk.

Ha szükséges, akkor további betűkészleteket ágyazhatunk a BASIC programunkba. Az MMBasic főmvert tartalmazó zip fájl része egy részletesebb leírás a saját fontok létrehozásához. A zip file tartalmaz néhány további betűkészletet is, amit tetszés szerint programunkba építhetünk, köztük a szimbólumokat tartalmazó (Dingbats) készletet, amivel könnyen hozhatók létre a képernyőn ikonok, stb. Ezek a fontok pontosan úgy használhatók, mint a beépített betűkészlet. (pl. kiválaszthatók a FONT parancs alkalmazásával, vagy megadhatók a TEXT parancs paramétereként).

A beágyazott fontok formátuma:

```

DefineFont #Nbr
    hex [[ hex[...]]
    hex [[ hex[...]]
END DefineFont

```

Vagyis a "DefineFont" kulcsszóval kell kezdődnie, amit a font száma követ (amit megelőzhet az opcionális # karakter). 1 és 16 között bármelyik szám megadható. Az ezt követő karakterdefiníciók 8 digitos hexadecimális szavak, amelyeket egy vagy több szóközzel vagy újsor karakterrel kell elválasztani egymástól. A fontdefiníció végét az "End DefineFont " kulcsszó jelzi.

Mikor a BASIC programot a programmemóriába mentjük, az MMBasic megkeresi a programban a beágyazott betűkészleteket, és a font táblázatba ezeket beilleszti. A program futása alatt ezeket a beágyazott fontdefiníciókat átlépi, vagyis ezek a program bármelyik részén elhelyezhetők.

A beágyazott fontokat a LIBRARY területén is tárolhatjuk. Ilyenkor ezek a fontok hozzáadódnak az MMBasic-hez, és bármikor használhatók a programban. Az MMBasic elhagyja a font definíciókban szereplő hex kódokat (mivel már nem kellenek), és bináris alakjukat tárolva, jelentős memóriamegtakarítást eredményez. A LIBRARY parancs részletes leírása ezen kézikönyv korábbi fejezetében található.

Az MMBasic alapértelmezett betűkészlete a font #1, azonban ez könnyen módosítható a FONT parancs alkalmazásával:

```

FONT font-number, scaling

```

Ahol 'font-number' egy szám, amit opcionálisan megelőzhet a hash (#) karakter. 'scaling' megadható (opcionális) és egy szám 1-15 között. A fontméret meg lesz szorozva ezzel a skálafaktorral, így a megjelenített karakterek szélesebbek és magasabbak lehetnek. Például, ha a 'scaling' értéke 2, akkor az megduplázza a karakterek magasságát és szélességét. Ha nem szerepeltetjük, akkor alapértelmezett értéke 1 (nincs nagyítás).

Rajzoló parancsok

A legtöbb rajzoló parancsnak vannak opcionálisan megadható paraméterei. Ezek a parancsok végéről elhagyhatók, vagy két vesszővel jelezzük a hiányzó paramétert. Például a LINE parancs ötödik paramétere opcionális, így a formátum:

```

LINE 0, 0, 100, 100, , rgb(red)

```

Az opcionális paramétereket a továbbiakban dőlt betűvel jelöljük. Például: *font*.

A következő parancsokban C jelöli a rajzvonala színét, és alapértelmezés szerint az aktuális előtérszín (foreground colour) értékét kapja. FILL a kitöltő szín, alapértelmezés szerint a -1 értéket kapja, ami azt jelzi, hogy nem használjuk a kitöltést. A rajzoló parancsok:

- **CLS C**
A kijelzőt a C-vel megadott színre törli. Ha C-t nem adjuk meg akkor az aktuális alapértelmezés szerinti háttérszínt használjuk.
- **PIXEL X, Y, C**
Egy pixel kigyújtása C színre. Ha C-t nem adjuk meg akkor az aktuális alapértelmezés szerinti előtér színt használjuk.
- **LINE X1, Y1, X2, Y2, LW, C**
Vonalhúzás X1,Y1 kezdőpontból X2,Y2 pontba C színnel.
LW a vonal szélessége, és csak vízszintes vagy függőleges vonalak esetén érvényes. Ha nem adjuk meg, vagy átlós vonalat húzunk, akkor az értéke: 1 pixel.
- **BOX X1, Y1, W, H, LW, C, FILL**
Doboz rajzolása X1,Y1 kezdőpontból. Szélessége W pixel, magassága H pixel.
LW a doboz körvonalainak a szélessége, akár 0 is lehet. Alapértelmezés szerinti értéke: 1 pixel.
- **RBOX X1, Y1, W, H, R, C, FILL**
Doboz rajzolása lekerekített sarkokkal. X1,Y1 kezdőpontból. Szélessége W pixel, magassága H pixel.
R a lekerekítés sugara a doboz sarkainál. Alapértelmezett értéke:10.
- **CIRCLE X, Y, R, LW, A, C, FILL**
Körrajzolás X,Y középponttal, R sugárral. LW a vonal szélessége, Nulla is lehet, alapértelmezett értéke:1.
A torzítási arány egy lebegőpontos szám, alapértelmezett értéke:1. Például, ha értéke: 0.5 akkor egy oválist rajzol, aminek szélessége fele a magasságának.

- `TEXT X, Y, STRING, JUSTIFICATION, FONT, SCALE, C, BC`
Szöveget jelenít meg X,Y kezdőponttól indulva. `JUSTIFICATION` egy vagy kétbetűs karakter jelölés, ahol az első betű a vízszintes szövegigazítást jelöli X irányban, és L, C vagy R lehet (LEFT, CENTER, RIGHT). A második betű a függőleges Y irányú elhelyezkedésre utal, és T, M vagy B lehet (TOP, MIDDLE, BOTTOM).
Az alapértelmezés szerinti igazítás: balra/fent. A Micromite Plus használhat további kódbetűket a szöveg forgatásához (ld. Micromite Plus Manual-t a részletekért). `FONT` és `SCALE` opcionálisak, és alapértelmezés szerint megegyeznek a `FONT` parancsban beállítottakkal. C a rajzoló szín, BC a háttérszín. Szintén opcionálisak, alapértelmezésük megegyezik a `COLOUR` parancsban megadottakkal.
- `GUI BITMAP X, Y, BITS, WIDTH, HEIGHT, SCALE, C, BC`
Egy bittérkép bitjeit jeleníti meg X,Y ponttól kezdődően. `HEIGHT` és `WIDTH` a bittérkép magassága és szélessége, ahogy majd megjelenik az LCD kijelzőn, alapértelmezés szerinti mérete: 8x8 pixel. `SCALE`, `C` és `BC` jelentése megegyezik a `TEXT` parancsnál leírtakkal.
A bittérkép lehet egész, vagy karakterfüzér változó, vagy állandó, és rajzoláskor a legfelső sorba kerülnek az első bájt bitjei (bit7 az első képpont bit6 a második, stb.), majd a második bájt, stb. Mikor a sor megtelt, a kitöltés a második sor elején folytatódik.

Példa

Példaként a következő program egy egyszerű digitális órát fog megjeleníteni egy ILI9341 alapú LCD kijelzőn. A program befejeződik, és visszatér a parancssorba, ha a kijelzőt megérintjük.

Először a kijelzőt és az érintésvezérlőt kell konfigurálni a már leírt módon, a következő parancsok kiadásával a parancssorból:

```
OPTION LCDPANEL ILI9341, L, 2, 23, 6
OPTION TOUCH 7, 15
```

Ezek beállítják a kijelzőt fekvő kijelzésre, és a 2, 23 és 6-os lábakat használják az LCD vezérléséhez, valamint a 7 és 15 lábakat az érintésvezérlőhöz (minden lábszám Micromite 28-ra vonatkozik). Ez a beállítás jó a következőkben ismertetendő Micromite LCD Backpack eszköz esetén, de az Ön konfigurációja ettől különbözhet.

A következőkben az érintésvezérlőt kalibrálni kell, kiadva az alábbi parancsot, és elvégezni a már előzőekben leírt kalibrációs eljárást:

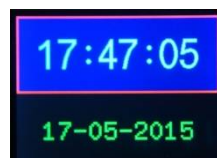
```
GUI CALIBRATE
```

Végül gépeljük be a programot, és futassuk:

```
CONST DBlue = RGB(0, 0, 128)           ' Sötétkék szín
COLOUR RGB(GREEN), RGB(BLACK)         ' Színeket beállítja
FONT 1, 3                             ' karaktereket beállítja
BOX 0, 0, MM.HRes-1, MM.VRes/2, 3, RGB(RED), DBlue
DO
  TEXT MM.HRes/2, MM.VRes/4, TIME$, CM, 1, 4, RGB(CYAN), DBlue
  TEXT MM.HRes/2, MM.VRes*3/4, DATE$, CM
  IF TOUCH(X) <> -1 THEN END
LOOP
```

A program a sötétkék szín állandóként történő megadásával, és az alapértelmezett elő- és háttérszínek, valamint a betűkészlet megadásával indul. Aztán rajzol egy dobozt vörös vonalakkal, sötétkék belsővel. Ez után egy ciklikus programhurok végrehajtása következik, amiben a következő történik:

1. Kijelzi az aktuális időt az előzőekben megrajzolt doboz belsejében. A kijelzett idő mind vízszintesen, mind függőlegesen a doboz közepére lesz igazítva. Megjegyezzük, hogy a `TEXT` parancs felülírja az alapértelmezett betűkészletet és színeket, a saját paramétereivel.
2. Kirajzolja a dátumot a kijelző alsó felének közepére igazítva. ebben az esetben a `TEXT` parancs az előzőekben beállított alapértelmezett betűkészletet és színeket használja.
3. Megvizsgálja, hogy megérintettük-e a képernyőt. Az érintést jelzi, amikor a `TOUCH(X)` függvény értéke nem -1, ekkor kilépünk a programból. A képernyőn valami hasonló jelenik meg (az itt használt betűtípus különböző):

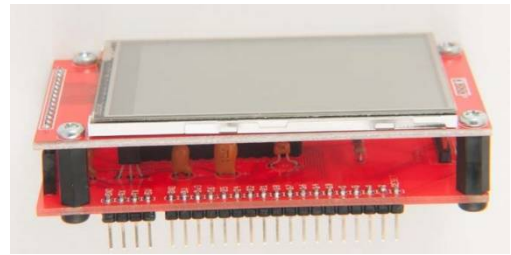


Micromite LCD Backpack modul

A Micromite LCD Backpack egy népszerű projekt, amiben a 28-lábú Micromite egy ILI9341 vezérlőt használó érintőképernyős LCD kijelzővel van kiegészítve.

A Micromite-ot tartalmazó NYÁK-ra lett rátervezve az LCD kijelző együttesen szendvicspanelként, és számos olyan megoldás alapja lehet, amelyhez egy könnyen programozható mikrovezérlőt és egy színes érintőképernyős LCD kijelzőt kell használni

A kevés alkatrészből álló Micromite LCD Backpack modul akár fél óra alatt összerakható.



A Micromite LCD Backpack leírása megtalálható: <http://geoffg.net/MicromiteBackpack.html>

Jelenleg négy elkészült megoldás is elérhető, és akár újraépíthető. Ezek:

Parkolást segítő: <http://geoffg.net/ParkingAssistant.html>

Hajó számítógép: <http://geoffg.net/BoatComputer2.html>

Szuper óra: <http://geoffg.net/SuperClock.html>

DDS Signal Generator: <http://geoffg.net/SignalGenerator.html>

A kézikönyv jelen fejezete azért készült, hogy egy „gyors kezdés”-t tegyen lehetővé azok számára, akik megépítik ezt a modult és gyorsan használatba akarják venni.

MMBasic programozása a PIC32-be

Ha a modul összeszerelése megtörtént, az első feladat a modul PIC32 mikrovezérlőjének a felprogramozása. Erre a legjobb megoldás a Microchip PICKit3 programozójának a felhasználása.

A programozás részletes leírása megtalálható::

http://geoffg.net/programming_pics.html

A backpack panelen van hat csatlakozópont ICSP felirattal jelölve, amihez a PICKit3 programozót a képen látható módon csatlakoztatjuk. A programozás előtt töltsük le a Micromite firmwaret tartalmazó állományt, aminek a neve: "Micromite Firmware V5.x" és megtalálható: <http://geoffg.net/micromite.html#Downloads> és ezt programozzuk be a PIC32-be.



Kapcsolódás a konzolhoz

A következő feladat a PC-n futó terminál emulátor csatlakoztatása a Micromite konzoljához. Ennek a teljes leírását már a kézikönyv egy előző részében (címe: „Konzol kapcsolat”) már leírtuk.

MMBasic konfigurálása

Ha konzol kapcsolódott, akkor megjelenik a képernyőn a ">" prompt. Ezután kell konfigurálni a Micromite-ot lehetővé téve az LCD kijelzőpanel és az érintőképernyő használatát, az LCD Backpack modulnál. Ehhez a következő parancsokat kell begépelni egymás után:

OPTION LCDPANEL ILI9341, L, 2, 23, 6

OPTION TOUCH 7, 15

Ezután kalibrálni kell az érintőképernyőt a következő begépelésével:

GUI CALIBRATE

A szereplő parancsok részletes leírása kézikönyv előző részeiben megtalálható.

BASIC program betöltése

A BASIC kód Micromite-ba való töltése a konzolon keresztül történik az AUTOSAVE vagy az XMODEM parancsok segítségével. Itt hivatkozunk a kézikönyv későbbi részeire, ahol ezek részletesen megtalálhatók.

Néhány program azt igényli, hogy a betűkészleteket külön töltsük be, és ezeket a könyvtárban tároljuk. A könyvtár egy speciális memóriaterület a Micromite-ban ahol a betűkészleteket és a program modulokat tárolhatjuk (ld. a "Speciális funkciók és a könyvtár" című fejezetet a kézikönyvben.). Az MMBasic tömöríti az adatokat a könyvtárban, ezért van hogy a programok igénylik a könyvtár használatát. Ha a betöltendő programnak van különálló betűkészlete, először ezt kell betölteni, és utána a LIBRARY SAVE parancsot kell használni. Ez a parancs a kódot átmásolja a könyvtárba, és törli ezt a program memóriából.

Ezután betölthető a fő BASIC program és elindítható a RUN paranccsal.

Speciális eszközök támogatása

A könnyebb alkalmazás érdekében a Micromite számos, általánosan használt periféria kezelőprogramját (driver-t) tartalmazza. Ezek:

- Infravörös távirányító adó és vevő
- A DS18B20 hőmérséklet-érzékelő és a DHT22 hőmérséklet-, és páratartalom érzékelő
- LCD kijelző modulok
- Numerikus billentyűzetek
- Segédtelepes óra IC-k
- Szervók
- Ultrahangos távolságérzékelő

Infravörös távirányító vevő dekódoló

Az alkalmazásban könnyedén megoldhatjuk egy külső infravörös távirányító használatát az IR parancs segítségével. Ha engedélyezve van ez a funkció és a háttérben fut, akkor lehetséges az éppen futó program megszakítása, ha megnyomunk egy gombot az infravörös távirányítón.

Működik bármelyik NEC vagy a Sony kompatibilis távirányítóval ideértve azokat is, amelyek kibővített üzeneteket állítanak elő. A legtöbb olcsó, programozható távirányító is ilyen protokoll szerinti üzenetet generál, és korszerű megoldásként használhatjuk Micromite alapú projektjeinkben. A NEC protokollt számos más gyártó is használja, köztük az Apple, a Pioneer, a Sanyo, az Akai és a Toshiba, így ezek a távirányítók is használhatók.

Az infravörös jel érzékeléséhez szükség van egy infravörös vevőre, amely csatlakozik az IR bemenetre (16. láb a 28-as Micromite-on) az ábrán látható módon. Az infravörös vevő érzékeli a jelet, demodulálja, és TTL feszültség szintű jelként beküldi az IR lábra. Az IR parancs az I/O láb irányát automatikusan beállítja.

A NEC távirányítók 38kHz frekvenciát és erre hangolt megfelelő vevőkészülékek használnak, mint a Vishay TSOP4838, Jaycar ZD1952 és Altronics Z1611A.

A Sony távirányítók 40kHz frekvenciát használnak, de vevőket erre a frekvenciára nehéz találni. Általában a 38kHz-es vevők működni fognak, de a maximális érzékenység eléréséhez 40kHz-es vevőt kell használni.

A dekóder beállításához a következő parancsot használjuk:

```
IR dev, key, interrupt
```

A dev változóba a készülék kódja, míg a key változóba a billentyűkód kerül eltárolásra. Minden gombnyomás hatására a megszakítás szubrutinra kerül a vezérlés. Az IR dekódolása a háttérben megtörténik, miközben a program tovább fut. Egy példa az IR dekóder használatára:

```
IR DevCode, KeyCode, IR_Int ' az IR dekodolo inditasa
```

```
DO
```

```
< a program torzse >
```

```
LOOP
```

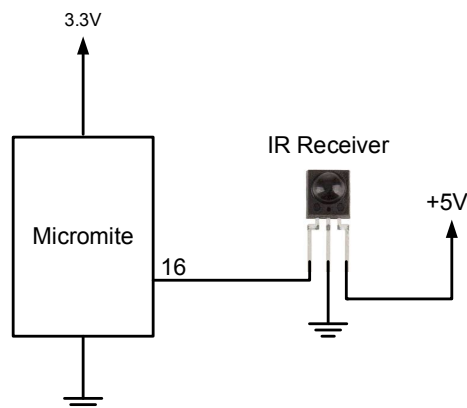
```
SUB IR_Int ' gombnyomast erzekeltunk
```

```
PRINT "Az eszkoz kod = " DevCode " Gombkod = " KeyCode
```

```
END SUB
```

Az IR távirányító különböző eszközöket képes kezelni (videomagnó, TV, stb). A program általában először az eszköz kódot vizsgálja, hogy ennek a programnak szól-e a parancs, és ha igen, a lenyomott gomb alapján elvégzi a műveletet. Mivel sok különböző eszköz és gombkód létezik, ezért ezek felderítésére a legjobb módszer a fenti program használata.

Az IR funkció ugyanazt az I/O lábat használja, mint a CPU SLEEP parancsnál az ébresztési jel. Ezek együttes használatára is mód van úgy, hogy a bejövő infravörös jel ébreszti a Micromite-ot, amely majd dekódolja az infravörös jelet. Ily módon lehet elérni, hogy egy akkumulátorról működő Micromite felébredjen egy infravörös jelre, végrehajtsa valamit a jel feldolgozása alapján, majd visszatérjen alvó módba.



A következő példa:

```
IR DevCode, KeyCode, IR_Int      ' IR dekodoló indul
DO
  CPU SLEEP                      ' mig nem jön jel, addig szundi
LOOP

SUB IR_Int                        ' gombnyomást érzekeltünk
  < valamit csinálunk amit a gombnyas aktivizál >
END SUB

                                ' visszateres a szundiba
```

Infravörös távvezérlő adó

Az IR SEND parancs segítségével továbbíthatunk egy 12 bites Sony infravörös távirányító jelet. Ennek eredeti célja két Micromite közti kommunikáció, de minden 12 bites kóddal működő egyéb Sony készülék közvetlen vezérlésére is használható. Megjegyzendő, hogy minden Sony termék megköveteli, hogy az üzenet háromszor küldjük el az üzenetek közötti 26msec késleltetéssel.

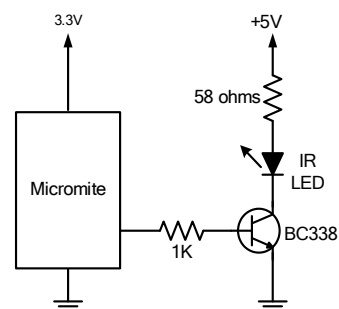
A Micromite firmware korábbi verzióiban ez a parancs az MMBasicba épült, de most már CSub modulként használható, ami pontosan ugyanaz. Lásd az *IRS-end.pdf* fájlt, amely a Micromite firmware zipfájlijában található az Embedded C Modules mappában.

A jobboldali ábra áramköre szemlélteti, hogy mire van szükség. A tranzisztort használjuk az infravörös led meghajtására, mert egy Micromite láb legfeljebb 10 mA árammal terhelhető. Ez biztosítja az 50 mA-es csúcsáramot a led meghajtására. Jelet a következő paranccsal küldhetünk:

```
IRSEND pin, dev, key
```

Itt pin a használt láb száma, a dev a készülék kódja és a key az eszköznek küldött parancskód. A Micromite bármelyik I/O lába használható, a beállítását az IRSEND parancs automatikusan elvégzi.

Az alkalmazott modulációs frekvencia 38 kHz, ez két egymással kommunikáló Micromite esetében pontosan belesik a leggyakrabban alkalmazott IR vevők optimális érzékenységu vételi sávjába.



Hőmérséklet mérése

A TEMPR() függvény segítségével mérhetünk hőmérsékletet egy DS18B20 hőmérséklet-érzékelővel. Ezt viszonylag olcsón lehet vásárolni, akár vízálló szonda változatban is.

A DS18B20 táplálható egy különálló 3V-5V-os táppal, vagy működhet a Micromite lábáról, mint ahogy ez jobb oldalon látható. Több érzékelő is használható, de mindegyikhez külön I/O lábra és felhúzó ellenállásra van szükség.

Aktuális hőmérsékletmérésre csupán a TEMPR() függvényt kell egy kifejezésben használni, például:

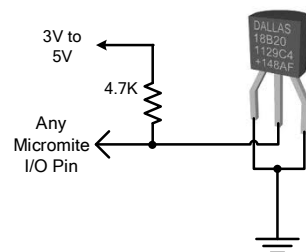
```
PRINT "Homerseklet: " TEMPR(pin)
```

Ahol 'pin' az alkalmazott I/O láb, beállítását az MMBasic automatikusan elvégzi.

A mért értéket °C-ban kapjuk, 0.25°C felbontással és ± 0,5 ° C pontossággal. Ha hiba történt a mérés során, akkor a visszaadott érték pontosan 1000 lesz.

A teljes méréshez szükséges idő 200 msec. A futó program megáll arra az időszakra, amíg a mérés megtörténik. Ez azt is jelenti, hogy a megszakítások le lesznek tiltva erre az időszakra. Ezt elkerülhetjük azzal, akkor külön elindíthatjuk a konverziót a TEMPR START paranccsal, majd később használjuk a TEMPR() függvényt a mért hőmérséklet kiolvasására. Amennyiben az érzékelő még nem végzett a méréssel, a TEMPR() függvény megvárja a befejezését. Például:

```
TEMPR START 15
< egyéb feladat vegzese >
PRINT "Homerseklet: " TEMPR(15)
```



Hőmérséklet és páratartalom mérése

A HUMID parancs fogja olvasni a hőmérsékletet és a páratartalmat egy DHT22-es hőmérséklet-, és páratartalomérzékelőről. Ez a szenzor RHT03 vagy AM2302 néven is kapható, de mindegyik kompatibilis és olcsón megvásárolható.

A Micromite firmware korábbi verzióiban ez a parancs az MMBasicba épült, de most már CSub modulként használható, ami pontosan ugyanaz. Lásd az *Humid.pdf* fájlt, amely a Micromite firmware zipfájljában található az Embedded C Modules mappában.

A DHT22 meghajtható 3.3V vagy 5V-al (5V ajánlott), valamint szükséges egy felhúzó ellenállás az adatláb és a táp között, ahogy ez az ábrán is látható. Ez különösen akkor fontos, ha hosszú kábelt (akár 20 méteres) használunk. Rövid kábelnél az ellenállás elhagyható, mivel a Micromite is rendelkezik egy belső gyenge felhúzó ellenállással.

A hőmérséklet és a páratartalom mérésére használjuk a HUMID parancsot három paraméterével:

```
HUMID pin, tVar, hVar
```

Ahol 'pin' a használt I/O láb. Bármelyik I/O lábat használhatjuk, de 5V-os táplálás esetén az csak 5V toleráns lehet. A beállítását az MMBasic automatikusan elvégzi.

'tVar' egy lebegőpontos változó, amiben a hőmérsékletet kapjuk vissza, és hasonlóan 'hVar' változóba kerül a mért páratartalom. A mért hőmérsékletet °C -ban kapjuk vissza, egytizedes felbontással (pl. 23.4), a relatív páratartalmat pedig százalékban (pl. 54,3). Például:

```
DIM FLOAT temp, humidity
HUMID pin, temp, humidity
PRINT "Hőmérséklet: " temp " Páratartalom(%): " humidity
```

RTC óra IC illesztése

Az RTC GETTIME parancssal könnyen kezelhetjük a PCF8563, DS1307, DS3231 vagy DS3232 típusú valós idejű órákat, valamint a velük kompatibilis eszközöket, mint pl. az M41T11. Ezek az integrált áramkörök népszerűek, olcsók, és képesek folyamatosan követni a pontos időt még a táplálás megszűnésekor is. Kompletต์ modulok beépített akkumulátort is tartalmazhatnak.

A PCF8563 és DS1307 havonta egy-két perces pontosságúak, míg a DS3231 és DS3232 áramkörök különösen precízek és pontosak, egy percen belül tartják a pontos időt évente.

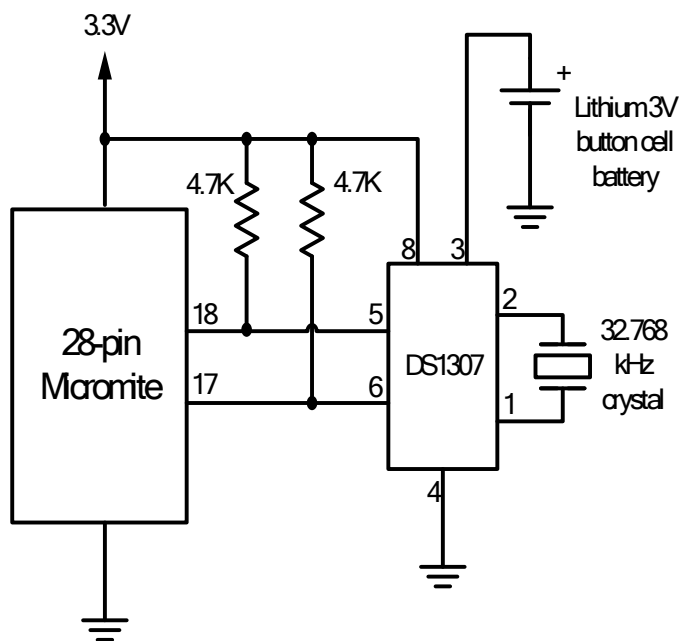
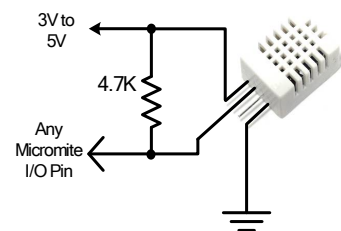
Az óra IC-k I²C buszos eszközök, emiatt össze kell kötni őket a Micromite I²C I/O lábaival. A jobb oldali ábra egy tipikus DS1307 óra IC bekötését mutatja. Más tokok bekötése is hasonló, de felhasználásuk előtt bekötésüket természetesen adatlapjukon ellenőrizni kell.

Mivel belső 100KΩ-os felhúzó ellenállások vannak az I²C I/O lábakon, ezért külön felhúzó ellenállásokat (mint az ábrán) sok esetben nem kell külön bekötni.

Az RTC IC első használata előtt be kell, be kell állítanunk az aktuális időt. Ezt az RTC SETTIME parancssal végezhetjük. Az RTC SETTIME formátuma: év, hónap, nap, óra, perc, másodperc. Ne feledjük, hogy az órát 24-órás formátumban kell megadni.

Például 2016. november 10. 4:00 óra:

```
RTC SETTIME 16,11,10,16,0,0
```



Az RTC GETTIME paranccsal könnyen megkaphatjuk az aktuális időt az óra IC-ből, majd ezzel frissíthetjük a Micromite belső óráját.

Normális esetben ezt a parancsot a program elején kell elhelyezni, hogy az idő beállítása bekapcsoláskor megtörténhessen.

A Micromite belső órája az MM28 és MM44-es típusoknál óránként akár két-három másodperccel is eltérhet, így a pontos időméréshez az RTC áramkört célszerű rendszeresen lekérdezni a SETTICK parancs segítségével és frissítenünk vele a pontatlan Micromite időmérést. Például így:

```
RTC GETTIME                                ' indulaskor az ido beallitasa
SETTICK 12 * 3600000, SetTime, 4          ' megszakitass 12 orankent

< a programunk >

SUB SetTime                                ' 12 orankent meghivott megszakitass
    RTC GETTIME                            ' ido frissitese
END SUB
```

Távolság mérése

Egy HC-SR04 ultrahangos érzékelővel és a DISTANCE() függvény segítségével lehet távolságot mérni az érzékelő és a célfelület között. A mérési tartomány 3 cm-től 3 méterig terjedhet. Úgy működik, hogy küld egy ultrahang pulzust és megméri a visszaérkezésig eltelt időt.

A Micromite firmware korábbi verzióiban ez a parancs az MMBasicba épült, de most már CSub modulként használható, ami pontosan ugyanaz. Lásd az *Distance.pdf* fájlt, amely a Micromite firmware zipfájljában található az Embedded C Modules mappában.

A Micromite-nál használt DISTANCE függvény:

```
d = DISTANCE(trig, echo)
```

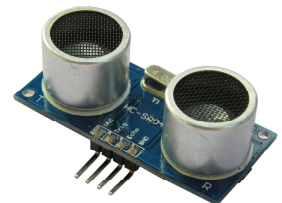
Ahol 'trig' az a láb, ahova az érzékelő „trig” kivezetése csatlakozik,

'echo' az a láb, ahova az érzékelő „echo” kivezetése csatlakozik,

3 kivezetésű eszköznél csak egy lábat kell bekötni.

A visszaadott érték a távolság centiméterben.

A láb konfigurálása automatikusan történik, de 5V toleráns lábnak kell lennie, mert a HC-SR04 5V-os eszköz.



Karakteres LCD kijelző

Az LCD parancs szöveget jelenít meg egy karakteres LCD modulon, minimális programozási munkával. Ez a parancs azoknál az LCD moduloknál fog működni, amelyeken KS0066, HD44780 vagy SPLC780 vezérlőt használnak, és 1, 2 vagy 4 soros a kijelzési képük. A karakteres LCD kijelzők könnyen beszerezhetők:

<https://shop.chipcad.hu/Welcoming/Default.aspx?scenarioID=360&pid=1630>

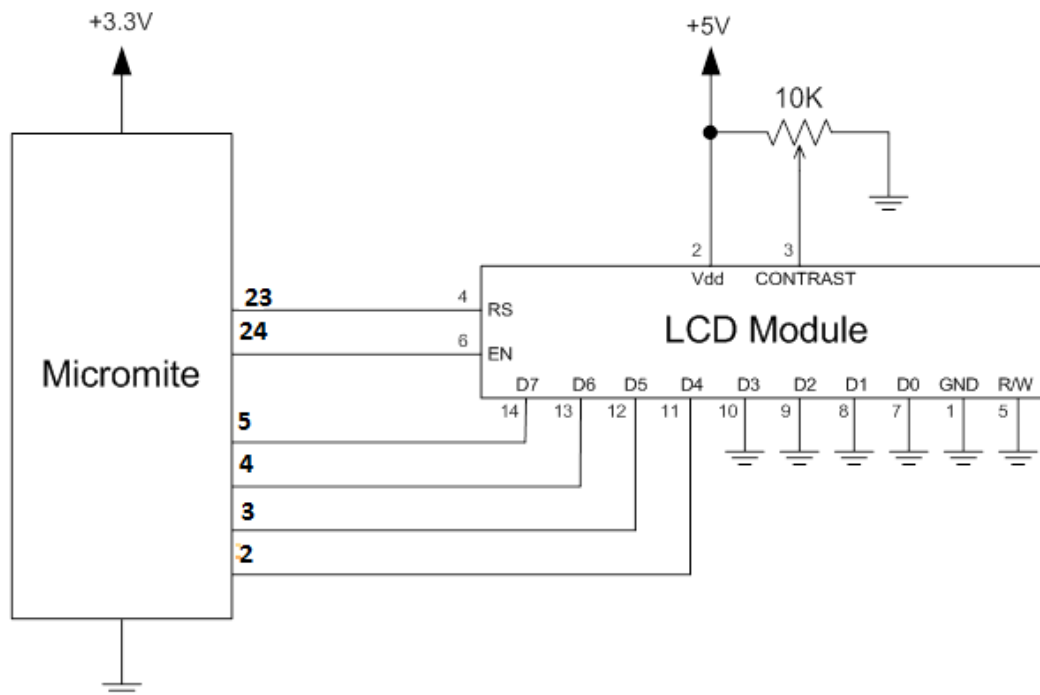
A modul inicializálása az LCD INIT paranccsal történik:

```
LCD INIT d4, d5, d6, d7, rs, en
```

A 'd4', 'd5', 'd6' és 'd7' konstansok a Micromite I/O lábainak a számai, amelyekkel a modul D4, D5, D6 és D7 kivezetéséhez csatlakoznak (a modul D0-D3 és R/W kivezetéseit le kell földelni). Az 'rs' a modul regiszterválasztás bemenetéhez csatlakozó láb (néha a neve CMD vagy DAT). Az 'en' a modul engedélyező lábára van kötve.

A Micromite bármelyik I/O lábát lehet használni, és nem kell az irányukat előre beállítani (az LCD parancs automatikusan megteszi ezt). A következő ábra egy tipikus bekötés:





Karakter megjelenítéséhez használjuk az LCD parancsot:

```
LCD line, pos, data$
```

Ahol a line a kijelző adott sorának száma (1-4), a pos a sor kezdőkaraktere, ahova az adatokat kell beírni (pl. az első sor 1 pozíciója 1). A data\$ az adatokat tartalmazó megjelenítendő karakterlánc az LCD-kijelzőn. A kijelzőn lévő karakterek felülíródnak.

A következőkben egy tipikus, a fenti ábrán látható beállítást mutatunk be: d4-d7 a Micromite 28 2-5 lábaira kapcsolódik, az rs a 23-as az en pedig a 24-es lábára:

```
LCD INIT 2, 3, 4, 5, 23, 24
LCD 1, 2, "Homerseklet:"
LCD 2, 6, STR$(TEMPR(15)) ' DS18B20 a 15-os labra van kotve
```

Megjegyezzük, hogy a példában a TEMPR függvényt használjuk hőmérséklet mérésére, de az ábrán a hőmérő bekötése nem szerepel.

Billentyűzet illesztése

A billentyűzet segítségével egyszerűbben vihetünk be adatokat Micromite alapú rendszerünkbe. A Micromite támogatja a mátrixba kötött 4x3-as, vagy 4x4-es billentyűzetet, figyelni és dekódolja a billentyűnyomásokat.

A gomb megnyomása megszakítást generál, az így elindított programrutin pedig feldolgozza. Egy 4x4 billentyűzetre példa a Digilent PMOD KYPAD.

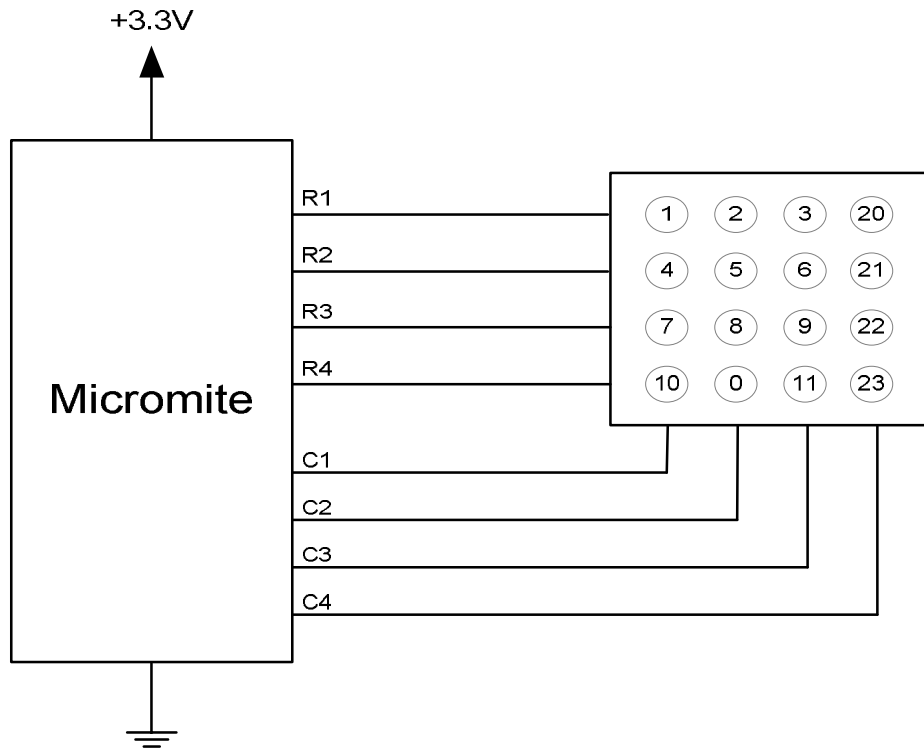
<https://shop.chipcad.hu/Welcome/Default.aspx?scenarioID=301&StockCode=DIG112#TabControl-2>

Billentyűzet funkció használatához használható parancs:

```
KEYPAD var, int, r1, r2, r3, r4, c1, c2, c3, c4
```

A 'var' változóban adja vissza a lenyomott billentyű kódját, az int a megszakítás kiszolgáló rutin neve, 'r1'-r4' a négy sor, 'c1'-c4' a négy oszlop bekötési pontjai a Micromite-on. A c4 csak akkor használt, ha 4x4-es billentyűzetet használunk. A Micromite bármelyik I/O lába használható, külön beállítani sem kell ezeket (a KEYPAD parancs ezt automatikusan megteszi).

A felismerés és dekódolás egy billentyű lenyomásakor a háttérben történik, a program fut tovább a parancs végrehajtása után. Amikor egy gombnyomást észlel, a var változó értéke tartalmazza a gomb sorszámát, ezután történik a megszakítás kiszolgáló rutin meghívása.



Például:

```

Keypad KeyCode, KP_Int, 2, 3, 4, 5, 21, 22, 23 ' 4x3 billentyuzet
DO
  < programtorzs >
LOOP
SUB KP_Int ' gombnyomast erzekeltunk
  PRINT "A lenyomott gomb = " KeyCode
END SUB

```

Kapcsolók, érintkező bemenetek

Gyakran van szükség arra, hogy nyomógombot vagy kapcsolót használjunk egy feladatban. Ezt könnyen megtehetjük, mivel minden bemenetre programmal konfigurálható, egy kb. 100 kΩ értékű, belső felhúzó ellenállás.

A kapcsolót a GND és a bemeneti láb közé kell kötnünk. Ha a kapcsoló nyitott, a felhúzó ellenállás miatt a lábon magas szint (3.3V) lesz, a kapcsolót zárva pedig alacsony szint (0V) mérhető.

A láb beállítása:

```

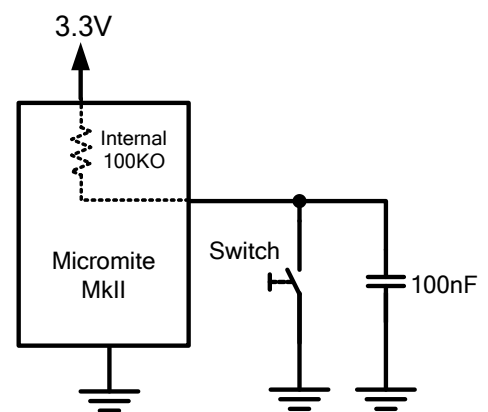
SETPIN pin, DIN, PULLUP

```

Ha olyan kapcsolónk van ami prellezik, akkor lehetséges, hogy egy gombnyomást többszöri gombnyomásként érzékelünk.

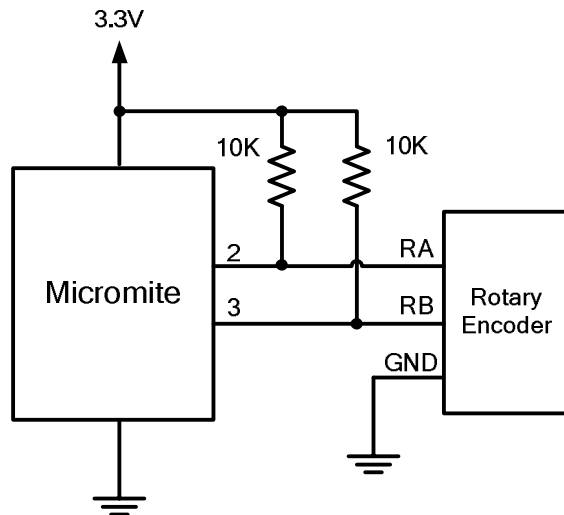
Ennek a problémának a kiküszöbölését prellmentesítésnek hívunk, amire két megoldás lehetséges.

- **Szoftveres prellmentesítés** esetén az első érzékelt állapotváltozás után 10-100 msec múlva ismét ellenőrizzük a kapcsoló állapotát, és ezt fogadjuk el.
- **Hardveres prellmentesítés** egy az érintkezővel párhuzamosan kötött 100nF kondenzátor beiktatásával lehetséges.



Elfordulás dekódoló

Az enkóder segítségével egyszerűen állítható be valamilyen paraméter egy mikrokontrolleres alkalmazásban. Az enkóder hasonló, mint egy potenciométer, de a rászertelt kezelógomb forgatásával Gray kódú jelsorozatot generál. A következő programrészlet mutatja, hogyan lehet dekódolni a kapott jeleket, és ennek segítségével frissíteni egy változót. Az eszköznek két kimenete (RA és RB), és egy GND kivezetése van. A kimenetekre az ábra szerinti felhúzó ellenállásokat kell kötni.



Az alábbi programrészlet dekódolja az enkóder kimenetét:

```
SETPIN 3, DIN                ' RB lab bemenet lesz
SETPIN 2, INTH, RInt         ' megszakítás, ha RA felfut

DO
  < main body of the program >
LOOP

SUB RInt                      ' megszakítás az enkóder kimenet dekódolására
  IF PIN(3) = 1 then
    Value = Value + 1        ' orajarassal egyező forgás
  ELSE
    Value = Value - 1        ' orajarassal ellentétes forgás
  ENDIF
END SUB
```

A programot TZAdvantage jóvoltából közölhetjük a The Back Shed Forumról. <http://www.thebackshed.com/forum/home.asp>

Ez a program azt feltételezi, hogy az enkóder a 2. és a 3. I/O lábához kapcsolódik, de szükség szerint bármely I/O lábra átköthető. A 'Value' változó értékét módosítja a megszakítási rutin, növeli vagy csökkenti, attól függően, hogy melyik irányban forgatjuk az enkóder tengelyét.

Szervo vezérlés

A szervo egy motor, beépített fogaskerekes áttétellel, és egy vezérlő rendszerrel, amely lehetővé teszi, hogy a tengely helyzete pontosan szabályozható legyen. A Micromite egyidejűleg akár öt szervo vezérlését is képes ellátni.

A normál szervók lehetővé teszik a tengely helyzetbe állását különböző szögekben, általában -90° és $+90^\circ$ fok között. Folyamatosan forgó szervók lehetővé teszik a tengely forgását és pozícióba állítását különböző sebességgel.

A szervo tengelyének helyzete egy 20 msec gyakorisággal ismétlődő impulzussal vezérelhető. Általában, ha az impulzus szélessége 0.8 msec, akkor a tengely pozíciója -90° , ha az impulzus szélessége 2.2 msec akkor $+90^\circ$, és 1.5 msec-es impulzus szélességnél a tengely (a rotor) középen áll meg. Gyártóként azonban ezek az értékek jelentősen eltérhetnek.

Méretüktől függően szervók elég erősek lehetnek, kényelmes lehetőséget nyújtanak akár nagyobb tömegek mozgására.

A legtöbb szervo nagy áramú 5V-os áramforrást igényel, melyhez két vezetékkel csatlakozik (piros +V és fekete a föld). A harmadik vezeték a vezérlőjel, amit a Micromite SERVO I/O lábához kell csatlakoztatni.

A Micromite-ban két szervo szabályozó egység van. Az elsővel három szervót vezérelhet, a másodikkal további két szervót. Az első vezérlőhöz csatlakozó, három szervót kezelő parancs:

```
SERVO 1, 1A, 1B, 1C
```

A második szervo vezérlőhöz kapcsolódó parancs:

```
SERVO 2, 2A, 2B
```

Ahol 1A, 1B, 2A, stb. a kívánt impulzus szélessége ezredmásodpercben minden kimenő csatornán. A kimeneti lábakat az előzőekben leírt lábdefiníciók alapján azonosíthatjuk, a jelölésük PWM 1A, PWM 1B, PWM 2A, stb. (PWM és SERVO parancsok szorosan összefüggnek, és ugyanazokat az I/O lábakat használjuk). Ha kevesebb szervót szeretnénk vezérelni, akkor hagyjuk ki a fel nem használtakat a parancsból, és az így megmaradt lábakat általános célú I/O lábként használhatjuk.

Az impulzus szélességét 0,005 msec-os pontossággal kell megadni, ez nagy felbontás. Például, az alábbi programrész a tengelyt középpólásba pozícionálja az 1A szervo csatornához csatlakozó szervónál:

```
SERVO 1, 1.525
```

Miután kiadtuk a SERVO parancsot, a Micromite egy folyamatos impulzussorozatot generál a háttérben, mindaddig, amíg egy másik SERVO parancsot nem kap, vagy egy STOP opcióval le nem állítjuk, ami megszünteti a kimeneti impulzusokat.

Egy másik példa: két szervo fog oda-vissza lengeni felváltva minden 5. másodpercben. Ehhez két szervót a PWM 1A és a PWM 1B kimenetekhez kell csatlakoztatni.

```
DO
```

```
    SERVO 1, 0.8, 2.2
```

```
    PAUSE 5000
```

```
    SERVO 1, 2.2, 0.8
```

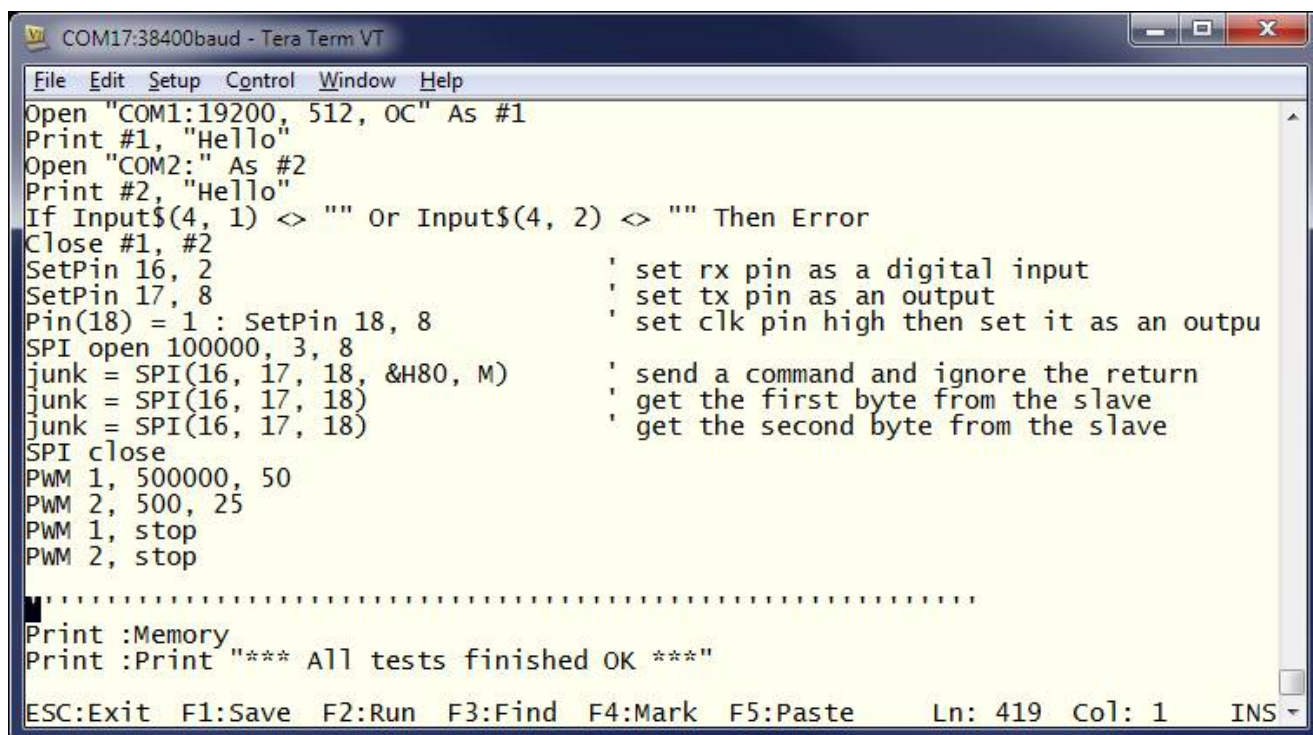
```
    PAUSE 5000
```

```
LOOP
```



Teljes képernyős szövegszerkesztő

A Micromite fontos, gyors fejlesztést lehetővé tevő eszköze a teljes képernyős szövegszerkesztő. Ez együttműködik minden VT100-kompatibilis terminál emulátorral (ajánlott PC szoftver a Tera Term).



```
File Edit Setup Control Window Help
Open "COM1:19200, 512, 0C" As #1
Print #1, "Hello"
Open "COM2:" As #2
Print #2, "Hello"
If Input$(4, 1) <> "" Or Input$(4, 2) <> "" Then Error
Close #1, #2
SetPin 16, 2           ' set rx pin as a digital input
SetPin 17, 8           ' set tx pin as an output
Pin(18) = 1 : SetPin 18, 8 ' set clk pin high then set it as an output
SPI open 100000, 3, 8
junk = SPI(16, 17, 18, &H80, M) ' send a command and ignore the return
junk = SPI(16, 17, 18)           ' get the first byte from the slave
junk = SPI(16, 17, 18)           ' get the second byte from the slave
SPI close
PWM 1, 500000, 50
PWM 2, 500, 25
PWM 1, stop
PWM 2, stop

.....
Print :Memory
Print :Print "*** All tests finished OK ***"

ESC:Exit F1:Save F2:Run F3:Find F4:Mark F5:Paste Ln: 419 Col: 1 INS
```

A teljes képernyős programszerkesztőt az EDIT parancssal indítjuk. A kurzor automatikusan a legutóbb szerkesztett helyre kerül. Esetleges programhiba okozta leállás esetén a kurzor a hibás sorra áll.

Ha már használtunk szövegszerkesztő programot, mint pl. a Notepad programot, akkor ismerős lesz a működése. Nyilakkal tudjuk mozgatni a kurzort a szövegben, a Home és az End billentyűvel a sor elejére vagy végére állhatunk. A Page Up és Page Down gombokkal lapozhatunk előre és hátra a programunkban. A Delete gombbal töröljük azt a karaktert, ahol a kurzor áll, backspace törli a kurzor előtti karakter. Az Insert billentyűvel választhatunk a beillesztő és felülíró mód között.

Az egyetlen szokatlan billentyűkombináció, hogy az egymás utáni két Home gombnyomás a program kezdetére, a kétszeres End gombnyomás pedig a program végére visz.

A képernyő alján a státusz sorban kiírja a különböző funkció gombokhoz tartozó szerkesztő parancsokat. Részletesebben ezek:

ESC	Ennek hatására a szerkesztő mindent félbeszakít, és visszatér parancssorba, a programmemória tartalma közben nem módosul. Ha megváltoztattuk a program szövegét, akkor megkérdezi, hogy valóban szeretnénk-e elhagyni a változtatásokat.
F1: SAVE	Elmenti a programot a programmemóriába és visszatér a parancssorba.
F2: RUN	Elmenti a programot a programmemóriába, és azonnal futtatja.
F3: FIND	Egy szöveget kér, amit keresni akarunk a programban. Ha enter-t ütünk, a kurzor az első megtalált szövegre áll.
SHIFT-F3	Ha már használtuk a kereső funkciót akkor segítségével többször is lehet keresni ugyanazt a szövegrészt.
F4: MARK	Ezt az alábbiakban részletesen ismertetjük.
F5: PASTE	Ezzel beilleszti (a kurzor helyére) azt a szöveget, amelyet előzőleg kivágtunk vagy másoltunk (lásd alább).

Ha megnyomta a jelölő gombot (F4) a szerkesztő átvált a jelölő módba. Ebben a módban arra használhatjuk a nyilakat, hogy kijelöljünk egy szövegrészt, amely inverz módban jelenik meg. Ekkor a kijelölt szöveget törölni,

kivágni vagy másolni lehet. Ebben az üzemmódban a státuszsor megváltozik, megjelenik ebben a módban a funkció gombok jelentése:

ESC	Kilép mark módból változtatás nélkül.
F4: CUT	Átmásolja a jelzett szöveget a vágólapra, és eltávolítja a programból.
F5: COPY	Vágólapra másolja a kijelölt szöveget.
DELETE	Törli a kijelölt szöveget, a vágólap tartalma változatlan.

Használhatunk vezérlő gombok helyett a fent felsorolt funkcióbillentyűket. Ezek a billentyűk:

LEFT	Ctrl-S	RIGHT	Ctrl-D	UP	Ctrl-E	DOWN	Ctrl-X
HOME	Ctrl-U	END	Ctrl-K	PageUp	Ctrl-P	PageDn	Ctrl-L
DEL	Ctrl-]	INSERT	Ctrl-N	F1	Ctrl-Q	F2	Ctrl-W
F3	Ctrl-R	ShiftF3	Ctrl-G	F4	Ctrl-T	F5	Ctrl-Y

A legjobb módja annak, hogy megtanuljuk a teljes képernyős szerkesztő kezelését, egyszerűen indítsuk el, majd kísérletezzünk vele!

Szövegszerkesztőnk nagyon hatékony eszköz programíráshoz. Az EDIT paranccsal szerkeszthetjük programunkat, ezután nyomjuk meg az F2 gombot, amivel elmentjük és futtatjuk azt. Ha a program hiba miatt leáll, akkor nyomjuk meg az F4 gombot, ami az EDIT parancsot futtatja, és a kurzor a hibás sorra áll. Ez a szerkesztés /fut-tatás/szerkesztés ciklus nagyon gyors, és kényelmes a tesztelés során.

Az OPTION BAUDRATE paranccsal a konzol sebességét akár 230400 bps-ra növelhetjük, ennél a nagyobb sebességnél az editor sokkal gyorsabban fogja újrarajzolni a képernyőt. Megbízható sebesség a 115200 bps.

Alaphelyzetben a szerkesztő azt várja, hogy a terminál emulátor 24 sor / 80 karakter széles üzemmódban működjön. Azonban mindkét beállítás megváltoztatható az OPTION DISPLAY paranccsal, hogy megfeleljen nem szabványos kijelzőknek.

Ne feledjük, hogy a terminál emulátor elveszítheti a pozícióját a szövegben több gyors gombnyomás (mint a fel és le nyilak) esetén. Ha ez megtörténik, akkor nyomjuk meg a HOME gombot kétszer, amely arra fogja kényszeríteni a szerkesztőt, hogy ugorjon a program kezdetére és rajzolja újra a képernyőt.

Színesen kódolt kijelzésű Editor

A szerkesztő képes színkód segítségével szerkeszteni a program kulcsszavait, a számokat és megjegyzéseket különféle színnel megjelenítve. Alap esetben a kimenet nem színkódolt, de ez a funkció bekapcsolható egy paranccsal:

```
OPTION COLOURCODE ON
```

és letiltható:

```
OPTION COLOURCODE OFF
```

A beállítás a memóriában tárolódik, és indításkor aktivizálódik.

Megjegyzés:

- Ehhez a funkcióhoz egy olyan terminál emulátort kell használni, amely képes értelmezni a megfelelő, a színeket kódoló szekvenciákat és azt megfelelően kezelni. Ez jól működik Tera Termmel azonban Putty esetén szükséges az alapértelmezett háttérszín fehérre megváltoztatni (Settings >> Colours >> Default Background >> Modify).
- Színjelöléshez a szerkesztő kimenetén igen sok extra karaktert kell küldeni a terminál emulátornak, és ez 38400 baud esetén lelassíthatja a képernyőfrissítést. Ha színes kódolást használunk, ajánlott, hogy nagyobb adatátviteli sebességet állítsunk be.

MM Edit fejlesztő környezet

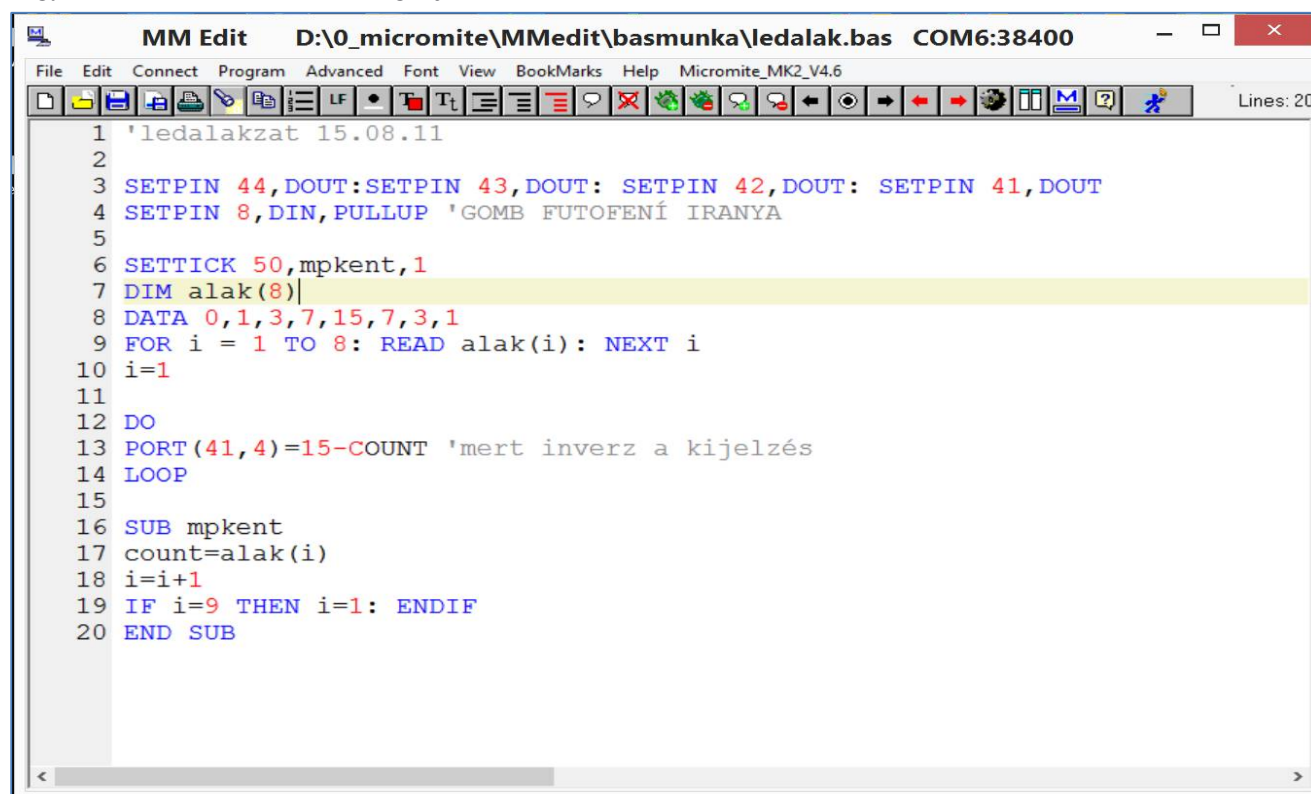
Az MMBasic programok fejlesztésére rendelkezésre áll egy integrált programfejlesztői fejlesztő környezet, az MM Edit: <http://www.c-com.com.au/MMedit.htm> Az MM Edit segítségével MM Basic programokat írhatunk, szerkeszthetünk és eltárolhatunk Windows vagy LINUX számítógépen. Az MM Edit a számítógép USB portján keresztül Micromite-tal kommunikál, és segítségével a BASIC programokat közvetlenül Micromite-ba tölthetjük és ott ellenőrzött körülmények között futtathatjuk.

A program telepítése a Windows programoknál megszokott módon történik. Nem ír a regisztrációs adatbázisba, és ezért létezik hordozható változata is. Installálás, és az az indítóikonjára történő rákattintás után, az ábrán látható képernyő fogad minket.

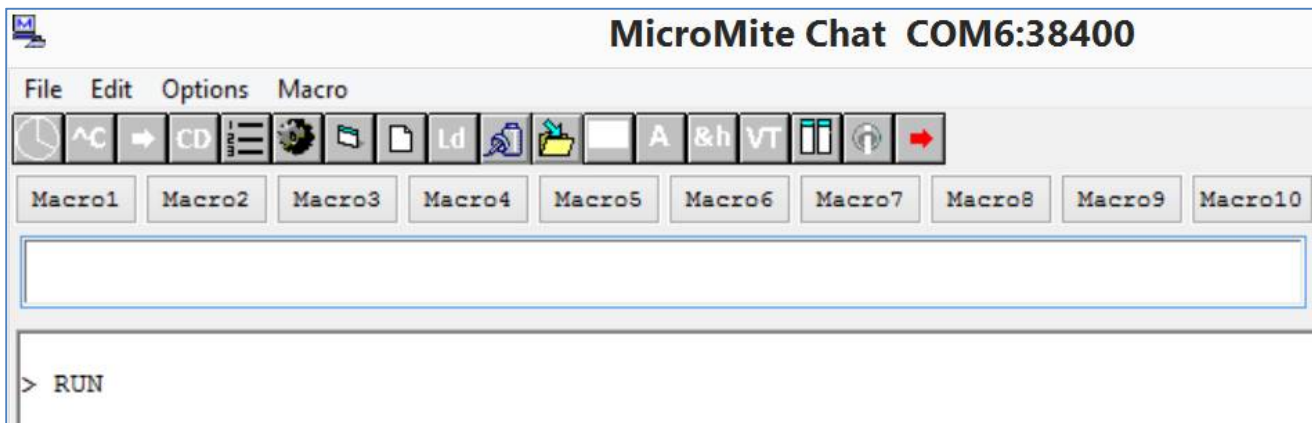
MM Edit: első lépések

A program részletesebb működésének megtanulásához rendelkezésre áll az *MMedit.pdf* kézikönyv, ezért a következőkben csak a legfontosabb jellemzőit mutatjuk be. Javasoljuk a kézikönyv letöltését és használatát: <http://www.c-com.com.au/stuff/MMedit.pdf>

Amint látható, számos szöveges legördülő menü található az ablak tetején. A gyorsabb elérés érdekében egy tekintélyes hosszúságú ikonsor is van a következő sorban, a gyakori parancsok gyors aktivizálásához. A képernyő nagy részét a szerkesztő ablak foglalja el.



1. Először a *Connect>New* legördülő menüben kell a megfelelő soros portot kiválasztani a felkínáltak közül, és megadni a 38400 bps adatátviteli sebességet.
2. Ezután a *File* menüpont segítségével, vagy egy meglévő (*File>Open*) programot töltünk be, vagy egy új programot írunk (*File>New*), vagy egy előre megírt sablon (template) felhasználásával készítjük el a programot (*File>New from Template...*). Ez utóbbi megoldást a következő részben ismertetjük.
3. A szerkesztő ablakban lévő programot módosíthatjuk az *Edit* menü parancsaival, és a szokásos szerkesztő billentyűkkel. Az így elkészített programot az ikonsor végén látható, kis futó emberkét ábrázoló *Load and Run current code* ikonra történő rákattintással tudjuk a Micromite-ba letölteni és azonnal futtatni. A letöltés közben megjelenik egy *Upload Progress* feliratú ablak, ahol a Micromite-al való kapcsolatfelvételt, és a letöltés menetét kísérhetjük figyelemmel.
4. Az ablak bezáródása után egy másik ablak nyílik meg, a neve: *MicroMite Chat*. Az ablak segítségével közvetlenül kommunikálhatunk a Micromite-al. Felül van a főmenü, alatta egy ikonsor, a makrósor, majd az egysoros beviteli szövegdoboz, ahonnan a beírt tartalmat elküldjük a Micromite-nak. Ez alatt van az általa küldött adatokat megjelenítő főablak. A főmenüvel, és a makrókkal kapcsolatos ismeretek a kézikönyvben találhatóak.



Ez az ablak három üzemmódban működhet, amit a menüsor A, &h, VT jelű gombjaira kattintva választhatunk ki, ezek sorrendben: ASCII mód, hexa mód, VT100 terminál emulátor mód.

- ASCII mód: Ez látszik az ábrán. Az egysoros szövegdobozba egy parancsot gépelünk be, és ezt Enter billentyűvel vagy a => jelű ikonra kattintással tudjuk a Micromite-nak elküldeni.
- A fel és le nyilakkal az utolsónak elküldött 30 sor közül választhatunk, szerkeszthetjük, majd ismét elküldhetjük
- Hexa mód: A HEX mód ugyanúgy működik, mint az ASCII mód, de a nagy kimeneti ablakban a küldött szöveg HEX alakban jelenik meg. Elsődlegesen a Micromite által küldött adatok vizsgálatára, pl. a VT100 vezérlő kódjainak elemzésére használható.
- VT100 mód: Egy VT100-as terminált emulál. NAGYON lassú, és könnyen puffer túlszordulás jöhet létre, ha gyorsan gépelünk. Az ikonsor melletti kezelőgombokkal lehet a paramétereket megváltoztatni. Részletek a kézikönyvben.

A VT100 módra váltva a felső ikonsor gombjai is változnak. A következőkben a leghasznosabb ASCII mód ikon-sorát ismertetjük, sorrendben balról-jobbra:

- Óra-jel: Elküldi a Micromite-nak a számítógép aktuális dátumát és idejét. Ilyenkor a Micromite-on program nem futhat.
- ^C: A futó programot leállító Ctrl+C billentyűkombinációt küld a Micromite-nak
- =>: A felső ablakban lévő egysoros szöveget küld el CR jellel végződve a Micromite-ba. Enter-t ütve a sorban lévő szöveg után is megtörténik az elküldés.
- A következő két gomb: Change directory és a Files a Micromite-nál nem használt.
- Fogaskerék: RUN a Micromite-ban lévő program futtatása.
- List: a Micromite-ban lévő program listázása.
- New: CLEAR és NEW parancs küldése a Micromite-nak, törölve a régit és előkészítve új program fogadására.
- A következő három gomb: Load, Paste, Toggle Capture a Micromite-nál nem használt.
- Fehér téglalap: Az ablak tartalmának törlése
- A, &h, VT: ASCII mód, hexa mód, VT100 terminál emulátor mód.
- File manager: Az MM file master ablak nyílik meg, amiben megvalósíthatjuk a Micromite és a PC közötti fájlvitelt.
- Í: itt nem használt
- Goto error line:

Sablonok (Fájl minta=Template) használata

Ha szeretnénk, hogy a programjaink egységes képet mutassanak, célszerű a sablonok (angolul: template) használata, amiből akár öt különféle sablon tárolható. Ha egy program több azonos sorral kezdődik, akkor az első alkalommal történő begépelés után célszerű sablonként bármilyen néven elmenteni.

Ha a fájl nevének a "DEFAULT.BT"-t adjuk, akkor *File>New* menüpont választásakor azonnal ez fog betöltődni.

Másik módszer: Elsőként el kell készíteni egy ilyen mintafájlt, és a *File>Open* paranccsal betölteni a szerkesztőbe. Ezután a *File>Save as Template* paranccsal elmenteni. Sajnos ahhoz, hogy ezt használni tudjuk, ki kell lépni, majd ismét belépni az MMedit programba. Ezután ezt a mintát a *File>New from Template...* paranccsal már használhatjuk. Maximum öt különböző mintát hozhatunk létre. A sablon megváltoztatható: megnyitás után módosítás, és azonos néven mentés. A sablonok a Data mappába vannak tárolva, az összes mappa helyét *Help>About ...* menüre kattintva láthatjuk.

Parancsok leírásának megjelenítése ablakban.

Ha egy parancs leírását akarjuk megnézni, akkor a kurzorral álljunk rá, és az ikonsáv végén látható kérdőjelet ábrázoló ikonra rákattintva, egy ablakban megjelenik a parancs angol nyelvű, rövid, összefoglaló leírása.

Könyvtár (Library)

Gyakran használt saját szubrutinokat, függvényeket egy külön Library területre tudjuk elmenteni. (File>Library).

Figyelem! Csak szubrutinokat és függvényeket menthetünk ilyen módon!

A szerkesztő ablakban kijelöljük a mentendő függvényt vagy szubrutint, majd megnyitjuk a File>Library menüt, és az „Add selected code to library” jelölésű gombra kattintunk. A mentettek programba való beillesztése a megnyílt ablak alapján magától értetődő. Ez a könyvtár szerkeszthető, a neve: library.bas és a Data mappában tárolódik.

Auto-Backup – Automatikus mentés

Lehetőség van arra, hogy a szerkesztés közbeni munkáinkat az MMedit automatikusan elmentse külön fájlként. A File>Preferences menüpontra kattintva megjelenő ablakban megkeressük az Auto-Backup Time felirat alatt lévő ablakot, itt írhatjuk be a periodikus mentés idejét percben. Ha az automatikus mentés ideje nem nulla, a programfájlokat a Folder ablakban megadott helyre menti ciklikusan. Mentésre kerül a dátum, az idő, majd a program neve: pl. 20140827_0738_programom.bas. Mentések letiltásához az időt nullára kell állítani.

Letöltés és futtatás esetén is megtörténik a mentés backup.bas néven az előbbieken megadott mappába.

Hibavadászat (debugging) segítése

Programok tesztelésekor fontos a program változóinak, aktuális állapotának megtekintése, a program futásának tetszőleges ponton történő megállítása (töréspont), valamint bármely programrész kizárása a futtatásból. Mindezt a forrásprogram módosításával tudjuk megvalósítani. *Minden változó értéke bármikor, akár a programban, akár a parancssorban megadott „? változónév” paranccsal kinyomtatható! (Fontos „?” után a betűköz megadása!)* Hibavadászatot támogató további eszközök:

- **Advanced>Mark selected line as comment** A program tetszőleges részét kommentnek jelölve, a program futásából ki tudjuk zárni.
- **Advanced>Uncomment selected lines** A fentiek fordított művelete.
- **Advanced>Mark selected lines as DEBUG** Hibakereséskor szokás a változókat kinyomtatni a PRINT utasítással. Ezeket a sorokat DEBUG szövegű kommenttel láthatjuk el.
- **Advanced>Remove DEBUG mark** a DEBUG kommentek eltávolítása.
- **Advanced>Mark DEBUG lines as comment** Teljes körű művelet az összes DEBUG sor kommentezésére.
- **Advanced>Uncomment DEBUG lines.** Teljes körű művelet az összes DEBUG sor visszaállítására.
- **TRON/TROFF, Nyomkövetés be/Nyomkövetés ki** parancsokat lehet elhelyezni a forrásprogramban tetszőleges szegmenseire. Az interpreter a konzol portra küldi az aktuálisan értelmezett sor számát [] zárójelbe zárva, így a nyomkövetés engedélyezésével és tiltásával követhetővé válik a programfutás menete.

Könnyű mozgás nagyméretű programban (könyvjelzők)

Nagyméretű programokban történő navigáció könyvjelzők segítségével lehetséges. (Bookmarks főmenü) Könnyen hozhatunk létre egy adott sorra mutató könyvjelzőket, törölhetjük is ezeket. A létrehozott könyvjelzők nem mentődnek el a programfájllal együtt.



Az MMedit felső ikonsorában folytonos vonallal bekeretezett két szélső gombbal is lehet lépkedni a könyvjelzők között körkörösén, a középső gombbal lehet létrehozni/törölni könyvjelzőt.

A szaggatott vonallal kijelölt két gombbal tudunk, egy felhasználó által definiált függvény vagy szubrutin nevére állítva a kurzort, elugrani annak a definíciójára, illetve onnan visszaugrani.

Fájlok méretének a csökkentése

Amikor a programunk mérete nagyobb lesz, az eszközmémoire mérete problémás lehet. Ahelyett, hogy a méretcsökkenés érdekében comment nélküli és így olvashatatlan, formázása nélküli kódot íránk, tartunk meg a jól formázott programkialakítást: számos kommenttel és az áttekintést segítő üres sorokkal. Ezután a fájl méretének csökkentéséhez letöltéskor fogunk tömöríteni.

Ez a tömörítés (angolul: crunch) elvégezhető a Program főmenüben található beállításokkal:

Remove blank lines - vegyük ki az üres sorokat,

Remove comments and blank lines, Távolítsuk el a megjegyzéseket, és az üres sorokat.

Remove indents and trailing spaces Bekezdéseket biztosító, és a sorok végén lévő betűközök eltávolítása

Crunch which does all of the above a fenti tömörítések végrehajtása egyszerre.

Program letöltése előtt célszerű megnézni *Program>Report Variable usage* jelentést, aminek elemzésével a nem használt függvényeket és szubrutinokat törölni tudjuk.

Változók definiálása és használata

Programokban változók formájában tárolunk számokat és szövegeket. A Micromite MMBasic három változótípus használatát engedi meg:

1. **Lebegőpontos szám**
Tárolásuk 32 bites változóban történik. Ezekben a változóban tizedesponttal elválasztott egész és tört-rész számokat tárolhatunk, pl. 45.386. Ezek igen nagy számok is lehetnek, de kisebb pontosságúak, amikor több mint hét számjegyből álló számot tárolunk, vagy módosítunk. A lebegőpontos változókat megadhatjuk a névhez hozzáírt "!" utótaggal (pl. i!, NBR!, stb). Ez az alapértelmezés szerint számábrázolási formátum akkor is, ha nem használjuk a "!" utótagot.
2. **64-bites előjeles egész szám**
Ezek pozitív vagy negatív számokat tárolhatnak, akár 19 decimális számjegyből állókat a pontosság elvesztése nélkül. Viszont törtrészt nem tartalmazhat. Ezt a formát megadhatjuk a névhez írt "%" utótag megadásával (pl. i%, NBR%, stb.). Megjegyezzük, hogy aritmetikai műveletek (különösen összeadás és a szorzás) során, nem történik meg az esetleges számábrázolási tartományból való kilépés (túl-vagy alulcsordulás) ellenőrzése. Ilyenkor az eredmény rossz lehet!
3. **Karakterfüzerek**
Ezek karaktersorozatokat tárolnak (pl., "Tom"). Minden karakter egy 8 bites értéként (0-255) van tárolva. A karakterfüzér változók végződhetnek a "\$" karakterrel (pl. name\$, s\$, stb) automatikusan jelölve a típusokat. A karakterfüzér akár 255 karakter hosszú lehet.

Megjegyezzük, hogy nem megengedett ugyanazt a változó nevet különböző típusú változóknál felhasználni

A legtöbb program lebegőpontos változókat használ, mivel ezek tipikus helyzetekben képesek megbirkózni a számokkal, jobban mintha egész és tört részeket használnánk.

Amikor egy egész számot használunk, azt feltételezzük, hogy ez egy olyan szám, aminek nincs tizedespontja vagy kitevője. Például 1234-et úgy értelmezzük, mint egy egész, míg az 1234.0 alakút lebegőpontos számként. A szöveg konstansokat mindig dupla idézőjelek közé zárjuk (pl. "karakterlánc").

OPTION DEFAULT

Egy változó használható utótag nélkül (pl. !, % or \$) és ilyenkor az MMBasic az alapértelmezett lebegőpontos típusként definiálja. Például a következő programsor egy lebegőpontos változót hoz létre:

```
Nbr = 1234
```

Azonban az alapértelmezett változótípus az OPTION DEFAULT paranccsal megváltoztatható. Például az, OPTION DEFAULT INTEGER parancs azt okozza, hogy minden utótag nélkül változó egész típusú lesz. Így a következő sorok egy egész típusú változót definiálnak:

```
OPTION DEFAULT INTEGER
Nbr = 1234
```

Az alapértelmezés állítható FLOAT (ez az alapértelmezett), INTEGER, STRING or NONE típusokra. Utóbbi esetben minden változó típusát meg kell adni, különben hibajelzést kapunk.

Az OPTION DEFAULT parancs bárhol elhelyezhető a programban, és bármikor módosítható, de a helyes gyakorlat az, hogy ezt a program elején helyezzük el és nem változtatjuk.

OPTION EXPLICIT

Alapértelmezésben MMBasic automatikusan létrehozza a változót akkor, amikor az először szerepel. Vagyis, a Nbr = 1234 utasítás létrehoz egy változót és értékét 1234-re állítja. Ez kényelmes, és gyors megoldás egyszerű programok esetén, de nagy programoknál rejtélyes, és nehezen megtalálható hibákhoz vezethet.

Például, a következő programrész harmadik sorában a Nbr változót elgépeltek Nr-re. Ennek következtében egy új változó jönne létre, nulla kezdőértékkel és Total értéke rossz lenne a gépelési hiba észlelése nélkül.

```
Nbr = 1234
Incr = 2
Total = Nr + Incr
```

Az OPTION EXPLICIT parancs viszont megtiltja, hogy az MMBasic automatikusan hozzon létre változókat. Helyette, a változók használata előtt deklarálni kell őket a DIM vagy LOCAL parancsokkal (ld. lentebb). Ha az OPTION EXPLICIT parancsot használjuk, akkor a hiba a program futtatáskor láthatóvá válik:

```

> LIST
OPTION EXPLICIT
DIM Nbr, Incr, Total
Nbr = 1234
Incr = 2
Total = Nr + Incr
>
> RUN
[5] Total = Nr + Incr
Error: Variable not declared
>

```

Az OPTION EXPLICIT használata előnyös, és jó programozói megoldás. Célszerű közvetlenül a program elején elhelyezni, változó deklarációk előtt.

DIM és LOCAL

A DIM és LOCAL parancsok használhatók változók deklarálására, típusuk beállítására, ha előzőleg az OPTION EXPLICIT parancsot alkalmazzuk.

A DIM parancs egy globális változót hoz létre, ami azt jelenti, hogy a program bármelyik részén használható, akár még a szubrutinokban és a függvényekben is. Ha azonban azt szeretnénk, hogy a deklaráció csak egy szubrutinban vagy egy függvényben legyen érvényes, akkor használjuk a LOCAL parancsot. A LOCAL és a DIM parancs megadási formája (szintaxisa) azonos.

Ha LOCAL-ként definiálunk egy változót ugyanazon névvel, mint egy globális változót akkor a globális változó rejtve lesz abban a szubrutinban vagy függvényben, ahol LOCAL-ként definiáltuk, és értéke csak ott belül lesz érvényes. Bármely LOCAL-ként definiált változó eltűnik, ha kilépünk a szubrutinból.

A legegyszerűbben a DIM és LOCAL arra használható, hogy egy vagy több változót definiáljunk az utótag vagy a hatályos OPTION DEFAULT alapján. Például:

```
DIM nbr%, str$
```

De arra is használható, hogy megadjuk egy vagy több változót egy adott típussal, utótag megadása nélkül:

```
DIM INTEGER nbr, nbr2, nbr3, etc
```

Ebben az esetben nbr, nbr2, nbr3, etc egészek lesznek. Mikor a változót a programban használjuk, nem kell a típusra mutató utótagot használni. Például a következő pont úgy működik, ahogy elvárjuk:

```
MyStr = "Hello"
```

A DIM és LOCAL parancsok használhatók a Microsoft szintaktika szerint is az "AS" kulcsszóval. Például:

```
DIM nbr AS INTEGER, str AS STRING
```

Ebben az esetben minden egyes változó típusának a beállítása egyedileg történik (nem csoportosan, mint amikor a típus kerül a változók listája elé).

A változóknak deklaráláskor értékek is adhatók. Például:

```

DIM INTEGER a = 5, b = 4, c = 3
DIM s$ = "World", i% = &FHHHHHHH
DIM str AS STRING = "Hello" + " " + s$

```

Megjegyezzük, hogy a változók értékadásakor kifejezéseket, vagy felhasználó által definiált függvényeket is használhatunk.

A DIM vagy LOCAL parancsokkal tömböket is definiálhatunk, és az előzőekben leírt szabályok alkalmazhatók, mikor tömböket definiálunk. Például, használható:

```
DIM INTEGER nbr(10), nbr2, nbr3(5,8), stb.
```

Amikor inicializáláskor egy tömb értékei szerepelnek, az értékeket tartalmazó vesszővel elválasztott listát zárójelbe kell tenni. Például:

```
DIM INTEGER nbr(5) = (12, 13, 14, 15, 16)
```

vagy

```
DIM days(7) AS STRING = ("Vas", "Het", "Ked", "Sze", "Csu", "Pen", "Szo")
```

CONST

Gyakran hasznos létrehozni egy konstans értéket tároló azonosítót, amelynek értékét nem lehet véletlenül megváltoztatni.

A CONST parancs használatával létrehozunk konstans azonosítót, ami úgy működik, mint egy változó, de az értéke nem változtatható meg. Például:

```
CONST Feszultseglab = 26
CONST MaxErtek = 2.4
```

Ezután azonosítókat használhatunk a program, ahol ez a megadott név többet jelent, mint egy egyszerű szám. Például:

```
IF PIN(Feszultseglab) > MaxErtek THEN Riasztas
```

Egy sorban több konstansot is megadhatunk:

```
CONST InputVoltagePin = 26, MaxValue = 2.4, MinValue = 1.5
```

Az alkalmazott érték lesz az állandó tartalma. Itt kifejezéseket, vagy felhasználó által definiált függvényeket is használhatunk.

A konstans típusát hozzárendelt értéke határozza meg, például a fenti MaxValue típusa lebegőpontos konstans, lesz, mert 2.4 egy lebegőpontos érték. A konstans típusa utótaggal (azaz !, % , \$) is beállítható.

Lebegőpontos és egész számok vegyítése

Az MMBasic automatikusan kezeli a számok konverzióját az egész számok és a lebegőpontosok között. Ha egy művelet keveri a lebegőpontos és egész típusokat (pl. PRINT A% + B!), az egész szám át lesz alakítva lebegőpontosra, mert az első szám lebegőpontos volt, és a művelet eredménye is lebegőpontos lesz. Ha a művelet operandusai egész számok, akkor az eredmény is egész lesz.

Az egyetlen kivétel a normál osztás ("/") amely a művelet mindkét oldalát lebegőpontosra konvertálja, és az eredmény is lebegőpontos lesz. Egész osztáshoz használjuk az egész osztás műveleti jelet: "\".

Függvények vagy lebegőpontos, vagy egész értékkel térnek vissza, a függvénytől függően. Például a PIN() egész értékkel tér vissza, ha a lábat digitális bemenetnek, és lebegőpontos értékkel, ha analóg bemenetnek konfiguráljuk.

Ha szükséges a lebegőpontos értéket egészre konfiguráljuk az INT() függvénnyel. Hogyha szükséges átalakítani egy egész számot lebegőpontosra, akkor elegendő az egész értéket egy lebegőpontos változóhoz rendelnünk, egy értékadás automatikusan konvertálni fogja azt.

64-bites előjel nélküli egészek

A Micromite 64-bites, előjeles, egész számokat támogat. Ez azt jelenti, hogy 63 bit hordozza az értéket és egy bit a szám előjelét (ez a legnagyobb helyi értékű bit). Egy 64-bites, előjel nélküli egész szám használata akkor lehetséges, ha aritmetikai műveletekben azt nem használjuk.

64-bites előjel nélküli számok létrehozhatók &H, &O vagy &B előtagokkal, és ez a szám egészsként tárolódik. Ezekkel csak korlátozott műveleteket végezhetünk! Ezek << (shift balra), >> (shift jobbra), AND (bitenkénti és), OR (bitenkénti vagy), XOR (bitenkénti kizáró vagy), = (egyenlő) és az <> (nem egyenlő). Az aritmetika műveletek, mint a +, -, stb. 64-bites előjel nélküli számokkal rossz eredményeket adhatnak.

A 64-bites értékek megjelenítéséhez a HEX\$(), OCT\$() vagy BIN\$() függvények használhatók.

Például a következő 64-bites előjel nélküli műveletek a várt eredménnyel térnek vissza:

```
X% = &HFFFF0000FFFF0044
Y% = &H800FFFFFFFFFFFFFFF
X% = X% AND Y%
PRINT HEX$(X%, 16)
```

A kijelzés: "800F0000FFFF0044"

I/O kivezetések használata

Digitális bemenetek

A digitális bemenet a legegyszerűbb típusú bemeneti konfiguráció. Ha a bemeneti feszültség nagyobb, mint 2,5V a logikai szint igaz lesz (számértéke 1), 0.65V alatt pedig hamis lesz (numerikus értéke 0). A bemenetek Schmitt-trigger jellegűek, így a bemeneti feszültség változása közben, a közbenső értékek esetén megmaradnak az előző logikai szinten. Az 5V-al jelölt lábak 5V toleránsak, azaz közvetlenül, feszültségejtő ellenállások nélkül kapcsolódhatnak olyan áramkörökhöz, melyek 5V tápfeszültségről működnek

A BASIC programunkban a digitális bemenetek állapotát a PIN() függvénnyel kaphatjuk meg. Például:

```
SETPIN 23, DIN
IF PIN(23) = 1 THEN PRINT "Magas"
```

A SETPIN függvény a 23-as lábat digitális bemenetnek konfigurálja, és a PIN() függvény tér vissza a láb állapotával (1-es érték jelenti a láb magas állapotát). Az IF parancs utáni THEN-t követő parancs fogja kiírni a szöveget, ha a láb állapota magas. Ha láb állapota alacsony, akkor a program az IF utáni sorral folytatódik.

A SETPIN parancsban használható pár lehetőség, például, hogy a bemenet belső ellenállással a tápfeszültségre vagy a földpontra kapcsolódjon. Ezek az úgynevezett "pullup" vagy "pulldown" ellenállások. Nagyon hasznosak, ha egy egyszerű kapcsolóval, vagy nyomógommbal csatlakozunk a bemenethez, mert az illesztéséhez szükséges külső ellenállást elhagyhatjuk.

Analóg bemenetek

Az ANALOG jelöléssel megadott bemeneti lábak feszültségmérésre használhatók. A mérhető feszültségtartomány 0-3.3V közötti, és szintén a PIN () függvénnyel jeleníthetjük meg az értékét. Például:

```
> SETPIN 23, AIN
> PRINT PIN(23)
2.345
>
```

Ha 3.3V feszültségnél nagyobb értéket kell mérnünk, akkor külső feszültségosztót kell használnunk, kis feszültségek méréséhez pedig külső erősítőket érdemes alkalmazni.

A mérésakor referenciaként az analóg táp láb feszültségét használjuk (28-as láb a 28-lábú verziónál, 17-es láb a 44-lábú verziónál), és feltételezzük, hogy a feszültség pontosan 3.3V. Ha ez a feszültség nem 3.3V (pl. telepes táplálás), akkor ezt korrigálhatjuk:

$$A = (\text{PIN}(x) / 3.3) * \text{PowerV}$$

ahol "PowerV" a feszültség az analóg táp lábon.

A feszültségmérés nagyon érzékeny az analóg Power és a Ground lábakon jelen levő zajra. A pontos és megismételhető feszültségmérések érdekében ügyelni kell paneltervezésnél, hogy gondosan elkülönítsük az analóg és digitális áramkörök táplálását, és biztosítsuk, hogy analóg tápegységünk zajmentes legyen.

Számláló bemenetek

A mikrovezérlő lábai közül a COUNT jelzésű bemenetek alkalmasak frekvencia, periódusidő mérésre vagy impulzusok számlálására. A következő programrészlet kiírja a 15. lábra kötött periodikus jel frekvenciáját:

```
> SETPIN 15, FIN
> PRINT PIN(15)
110374
>
```

Ebben az esetben a frekvencia 110.374 kHz.

Alapértelmezésben a frekvenciamérés kapuideje egy másodperc, ez idő alatt beérkező impulzusokat számolja meg, vagyis a mérések frissítési frekvenciája egy másodperc. Ha a SETPIN parancsnál egy harmadik paramétert is megadunk, akkor a kapuidőt 10 msec és 100000 msec között változtathatjuk.

Rövidebb kapuidő gyakoribb kiolvasást okoz, de pontatlanabb. A PIN() függvény mindig Hz-ben adja vissza a frekvenciát, a kapuidőtől függetlenül.

Például, az előző feladat 10 msec-os kapuidővel és kisebb pontossággal:

```
> SETPIN 15, FIN, 10
> PRINT PIN(15)
110300
>
```

10 Hz-nél alacsonyabb frekvenciák esetén előnyösebb periódusidő-mérést használni. Ilyenkor a bemenő jel két felfutó élével kapuzzuk a belső generátor msec felbontású jelét. A mért értéket a mért periódus alacsony-magas átmeneténél frissítjük. Így, ha mondjuk a periódusidő 100 másodperc, akkor ennyi idő kell, míg a PIN() függvény értéke frissül.

A COUNTING lábak bejövő impulzusok megszámlálására is alkalmasak. Amikor egy lábat így definiálunk (pl. SETPIN 15, CIN) a hozzá kapcsolódó belső számláló lenullázódik, majd lépteti a számlálót minden bejövő felfutó élű impulzus. A számláló ismét nullázható egy SETPIN paranccsal (függetlenül attól, hogy előzőleg már számlálóként konfiguráltuk).

A Micromite nagyon keskeny, akár 10 ns szélességű impulzusokat is képes számlálni. A mérhető frekvencia akár 800kHz is lehet, de általában a processzor terhelése (pl. soros vagy I²C kommunikáció) miatt inkább 300 kHz-el érdemes számolni.

Digitális kimenetek

Minden I/O lábat be lehet állítani szabványos digitális kimenetnek. Ez azt jelenti, hogy amikor egy kimeneti lábat 0-ra állítunk, akkor kimenet közel 0V, míg 1-be állítva azt, 3.3V feszültségű lesz. MMBasic utasításokkal: PIN (15) = 0, illetve PIN (15) = 1. A láb képes 10 mA áramot szolgáltatni, ami elég egy led vagy más logika meghajtásához.

A "OC" opció a SETPIN parancsban a kimeneti lábat nyitott kollektorosnak konfigurálja. Ez azt jelenti, hogy a kimeneti meghajtó a kimenetet nullára húzza, ha a kimenetet 0-ra állítjuk, de kimeneti 1-be állítás esetén az magas impedanciájú lesz. Egy külső felhúzó ellenállás alkalmazásával ez feszültségillesztésre alkalmas.

Impulzus szélesség moduláció (PWM)

A PWM (Pulse Width Modulation = impulzus szélesség moduláció) parancs lehetővé teszi, hogy a Micromite olyan négyszögjelet generáljon, aminek a magas-alacsony állapotának aránya, más néven a kitöltési tényezője (duty cycle) változtatható. Ezzel a kimeneti feszültség középértéke, és a vele táplált eszköz (pl. motor) árama változtatható, ami arányos a motor fordulatszámával és nyomatékával. PWM kimenet szervók meghajtására, de akár hangok létrehozására is használható.

Két PWM vezérlő van. Az elsőnek három kimenete van, a másodiknak kettő, azaz összesen öt PWM kimenet áll rendelkezésre. Mindkét controller működési frekvenciája egymástól függetlenül állítható a 20 Hz-500 kHz között, míg a kitöltési tényezők mind az öt kimenetnél 0% és 100% között állítható 0,1%-os felbontással, ha a frekvencia 25 kHz alatt van. 25 kHz felett a felbontás 1%-os.

A Micromite bekapcsolásakor, vagy a PWM OFF parancs használatakor a PWM kimenetek magas impedanciájú állapotba kerülnek. Tehát, ha azt szeretnénk, hogy a PWM kimenet alacsony legyen alapértelmezés szerint (ez a nulla teljesítmény a legtöbb alkalmazásban), akkor használjunk egy lábra kötött ellenállást, aminek másik végét a földre kötjük. Hasonlóképpen, ha azt szeretnénk, hogy az alapértelmezett PWM állapot magas legyen (teljes teljesítmény), akkor az ellenállás másik végét 3.3V-ra kötjük.

Megszakítások

Megszakítások használatával olyan eseményeket tudunk feldolgozni, amelyek bekövetkezésének időpontja előre nem ismert.

Erre példa az, amikor a felhasználó megnyom egy gombot. Bár lehet olyan programot írni, ami ciklikusan vizsgálja a gomb állapotát, de a gomb megszakítással történő figyelése sokkal egyszerűbb.

Beállíthatjuk, hogy a gomb megnyomása megszakítást okozzon. Ilyenkor az éppen futó program futása megszakad és egy megszakítást kiszolgáló programrész hajtódik végre, majd a program a megszakított résztől tovább folytatódik.

Bármelyik láb, amelyik használható digitális bemenetként, konfigurálhatjuk megszakítást kérő bemeneteknek. A SETPIN parancs segítségével lehet beállítani a megszakítást használó bemeneteket, maximum tízet. Beállíthatjuk, hogy a megszakítás a lábon megjelenő jel fel, vagy lefutó élére történjen. Bekövetkezésekor egy azonnali ugrás történik egy megadott címkével megjelölt szubrutinra. A megszakításból való visszatérés a megszakítási alprogram végén lévő END SUB vagy EXIT SUB utasítások végrehajtásával történik. Megjegyezzük, hogy ezek

a függvények paraméterátadásra nem alkalmasak, de a megszakítási alprogramban a GOTO, GOSUB parancsok és más alprogramok hívása is használható.

Ha két vagy több megszakítás történik ugyanabban az időben, a feldolgozás abban a sorrendben történik, ahogy a SETPIN paranccsal definiáltuk. Egy megszakítás kiszolgálása során minden más megszakítás tiltva van. A megszakításra definiált láb egyébként a megszokott módon, a PIN () függvénnyel kezelhető.

Megszakítások bármikor kiszolgálásra kerülnek, de az INPUT utasítások végrehajtásakor nem lesznek felismerve. Néhány hosszú, hardverrel kapcsolatos műveletnél (pl. TEMPR() függvény hívása) is igaz ez, és csak akkor lehet ezeket kiszolgálni, ha megszakítást kérő esemény még fennáll.

Amikor megszakításokat használunk, a főprogramot nem érinti a megszakítás tevékenysége, kivéve, ha olyan változót használunk a főprogramban, amit a megszakítási alprogram esetleg megváltoztat.

Az MMBasic legtöbb programjánál a megszakításkérésre adott válasz 30 microsec alatt megtörténik. A főprogram lelassulásának megelőzésére a megszakítások legyenek rövidek, és gyorsan fejeződjenek be. Fontos, hogy ha nincs szükség rá, tiltsuk le a megszakítást, mert a háttérben futó megszakítások misztikus hibákat okozhatnak.

Ne feledjük, hogy a régi szabvány szerint egy megszakítás egy sorszámmal vagy címkével jelölt sorra ugrik, amit az MMBasic is támogat. Egy szubrutinhívás tisztább, egyszerűbb, és a dokumentálása is könnyebb, ezért ezt kell előnyben részesíteni. Ha a megszakítás egy sorszámmal vagy címkére mutat, a megszakítás-kiszolgálás végét egy IRETURN paranccsal kell lezárni, aminek hatására a vezérlés visszatér a főprogramra.

Időzítés

Az MMBasic számos tulajdonsága teszi lehetővé időzítést igénylő események kezelését. Van egy belső órája, aminek segítségével az aktuális dátumot és az időt a DATE\$ és TIME\$ függvényekkel lekérdezhetjük, illetve módosíthatjuk. A naptár nulláról indul a Micromite első bekapcsolásakor, de egy valós idejű óra IC (pl. PCF8563) segítségével, az aktuális idő mindig betölthető.

A PAUSE parancs a program végrehajtását felfüggeszti a megadott számú milliszekundumig.

Például egy 12 msec széles impulzus létrehozása:

```
SETPIN 4, DOUT
PIN(4) = 1
PAUSE 12
PIN(4) = 0
```

A PULSE parancssal is létrehozhatunk nagyon rövid (20 microsec), vagy igen hosszú, több napig tartó impulzust, ami természetesen a háttérben fut, míg a főprogram változatlan sebességgel fut tovább.

Egy másik hasznos parancs a TIMER, amely úgy működik, mint egy stopper. Beállíthatunk bármilyen értéket (általában nullát), és felfele számol minden ezredmásodpercben.

Időzítést használ a SETTICK parancs is. Ez a parancs megszakítást generál periodikusan (adott ezred-másodpercenként). Úgy tekinthetjük, mint egy ütemadó.

Például, a következő kódrészlet másodpercenként kiírja az aktuális időt és a 2. láb feszültségét. Közben futhat a főprogram, amely egyéb tevékenységet végez:

```
SETPIN 2, AIN
SETTICK 1000, DOINT
DO
    ' a fő programhurok
LOOP

SUB DOINT ' periodikus idozito megszakitas
    PRINT TIME$, PIN(2)
END SUB
```

A második sor beállítja a megszakítás ütemét (angolul: tick), ennek első paramétere a megszakítás gyakorisága (1000 msec), a második pedig a megszakítást kiszolgáló rutin kezdő címe (DOINT). Minden másodpercben (azaz 1000 msec-ként) a fő feldolgozási ciklus megszakad és a DOINT címkénél kezdődő szubrutin kerül végrehajtásra.

Akár négy "tick" megszakítást állíthatunk be egy időben. Ezen a megszakítások prioritása a legalacsonyabb.

A gyártási tűrések miatt a Micromite belső órájának a pontossága változhat egy kicsit és a hőmérséklettől is függ. Ennek kiegyenlítésére az OPTION CLOCKTRIM parancssal lehet finoman állítani az órát, hogy pontosabb legyen.

Definiált szubrutinok és függvények

Gyakran használt tevékenységeket tetszőleges helyen aktivizálható alprogramok (szubrutinok), és függvények alkalmazásával jobb programszervezést, és könnyű módosíthatóságot tudunk megvalósítani. A megadott szubrutin vagy függvény pusztán csak egy programrész, amely meghívható bárhol a programon belül, és olyan mintha az MMBasic beépített parancsait kiterjesztenénk.

Tegyük fel például, hogy azt szeretnénk egy FLASH nevű parancsot létrehozni, ami a 2-es lábra kötött ledet egyszer megvillantja. Ennek megadása:

```
SUB FLASH
  SETPIN 2, DOUT
  PIN(2) = 1
  PAUSE 100
  PIN(2) = 0
END SUB
```

Ezután a programban már használhatjuk a ledvillogtató FLASH parancsot, például:

```
IF A <= B THEN FLASH
```

Ha a FLASH alprogram a programmemóriában van, akkor parancssorból is használható, mint minden más MMBasic parancs. A FLASH-alprogram megadása bárhol lehet a programban, de általában ezeket célszerű a program elején vagy a végén elhelyezni. Ha futás közben az interpreter ilyen definícióhoz ér, akkor azt egyszerűen átlépi.

Szubrutinok paraméterei (argumentumai)

A definiált szubrutinoknak paraméterei is lehetnek. Ez a következőképpen néz ki:

```
SUB MYSUB (arg1, arg2$, arg3)
  <statements>
  <statements>
END SUB
```

Amikor meghívjuk az alprogramot, hozzá lehet rendelni konkrét értékeket. Például:

```
MYSUB 23, "Cat", 55
```

A szubrutin belsejében szereplő arg1 értéke 23, arg2\$ értéke "Cat", stb. lesz. Az argumentumok úgy viselkednek, mint a közönséges változók, csak a szubrutin belsejében léteznek, és a szubrutin végrehajtása után eltűnnek. Használhatunk ugyan a főprogramban azonos nevű változókat, ezek a nevük azonosságának ellenére mások lesznek, de ez megnehezíti hibakeresést.

Amikor meghívunk egy szubrutint, használhatunk a definiáltnál kevesebb paramétert. Például:

```
MYSUB 23
```

Ebben az esetben a hiányzó értékek nulla, vagy egy üres karakterlánc fogja helyettesíteni. Például, a fenti esetben arg2\$ értéke "" és arg3 értéke nulla lesz. Ez lehetővé teszi, hogy legyenek opcionális értékek és, ha az értéket a hívó nem támogatja, akkor csinálhatunk valami speciális műveletet.

Akkor is igaz, ha kihagyunk egy értéket a lista közepén. Például:

```
MYSUB 23, , 55
```

Ekkor arg2\$ tartalma "" lesz.

Ahelyett, hogy típust jelölő utótagot (pl. \$ a arg2\$-nél), használnánk, alkalmazzuk az utótag AS <típus> definíciót a szubrutin paramétereinél, és ekkor megadott típusként fog szerepelni, akkor is, ha az utótagot nem használjuk. Például:

```
SUB MYSUB (arg1, arg2 AS STRING, arg3)
  IF arg2 = "Cat" THEN ...
END SUB
```

Ha nem adjuk meg argumentum típusát, akkor az alapértelmezés szerint lebegőpontos típusú lesz, vagy olyan amit az OPTION DEFAULT opcióban definiáltunk. Továbbá, ha használjuk az OPTION DEFAULT NONE beállítást, akkor a definiálást a szubrutinokban vagy a függvényekben is el kell végezni.

Helyi változók

A szubrutinon belül sokszor szükséges használnunk átmeneti változókat. Hordozható kód írásakor nem célszerű olyan neveket használni, ami azonossága miatt ütközhet a főprogramban használt változónevekkel. Ezért lehet definiálni a szubrutinon belül használható, ún. belső változókat (lokális). Ezt a LOCAL parancs segítségével végezzük el.

Például, a FLASH szubrutinban adjuk meg paraméterként azt, hogy hányszor (nbr) gyulladjon ki a led.

```
SUB FLASH ( nbr )
  LOCAL INTEGER count
  SETPIN 2, DOUT
  FOR count = 1 TO nbr
    PIN(2) = 1
    PAUSE 100
    PIN(2) = 0
    PAUSE 150
  NEXT count
END SUB
```

A számláló változót (count) a szubrutinon belül definiáljuk és csak a rutinon belül használható, és eltűnik, amikor kilépünk a szubrutinból. Persze lehet egy count változó a főprogramban is, de ez más, mint a szubrutin belsejében definiált és használt count nevű változó.

Ha nem deklaráljuk ezt a változót a szubrutinban és az OPTION EXPLICIT beállítás nem aktív, akkor létrejön ez a változó a szubrutinban történő használatkor és a főprogramban is látható lesz normál változóként.

Használhatunk lokális változókat a GOSUB hívásokban. Például:

```
GOSUB MySub
...
MySub:
  LOCAL X, Y
  FOR X = 1 TO ...
    FOR Y = 5 TO ...
      <statements>
    RETURN
```

Az X és Y változók csak addig érvényesek, amíg el nem érjük a RETURN utasítást, és különböznek a főprogramban használt esetleg ugyanilyen nevű változóktól.

Definiált függvények

A definiált függvények hasonlóak a definiált szubrutinokhoz, azzal a kivétellel, hogy az egyetlen visszaadott értéket a függvény neve hordozza. Például, két érték esetén a nagyobbikat akarjuk visszakapni a függvény nevében:

```
FUNCTION Max(a, b)
  IF a > b THEN
    Max = a
  ELSE
    Max = b
  ENDIF
END FUNCTION
```

A függvényt így használhatjuk:

```
SETPIN 1, AIN : SETPIN 2, AIN
PRINT "A nagyobb feszultseg: " Max(PIN(1), PIN(2))
```

A paraméterekre vonatkozó szabályok hasonlóak a szubrutinoknál megadottakhoz. Az egyetlen különbség az, hogy zárójelben kell megadni a paraméterek listáját, ha meghívunk egy függvényt (szubrutinoknál ez opcionális).

Visszatérési érték megadása úgy történik, hogy a függvény neve szerepel egy értékadási utasítás bal oldalán. Ha a függvény nevét \$, % vagy ! karakterrel zárjuk, akkor a függvény visszatérési értéke is ilyen típusú lesz. A típust megadhatjuk, az AS <típus> kifejezés szerepeltetésével is. Például:

```
FUNCTION Max(a, b) AS INTEGER
```

Ami ugyanaz:

```
FUNCTION Max%(a, b)
```

A függvény neve a függvényben a megadott típusú szabványos változóként működik.

Egy másik példa, hogyan formázzuk az időt 24 órás kijelzés helyett AM/ PM formátumban:

```
FUNCTION MyTime$(hours, minutes)
    LOCAL h
    h = hours
    IF hours > 12 THEN h = h - 12
    MyTime$ = STR$(h) + ":" + STR$(minutes)
    IF hours <= 12 THEN
        MyTime$ = MyTime$ + "AM"
    ELSE
        MyTime$ = MyTime$ + "PM"
    ENDIF
END FUNCTION
```

Mint látható, a függvény nevét (MyTime\$) mint egy közönséges helyi változót használjuk a függvény belsejében. A példa azt is mutatja, hogyan történik a lokális változók használata a szubrutinokhoz hasonló módon a függvényekben.

Név szerinti paraméterátadás

Ha egy egyszerű változót (nem kifejezést) használunk, mint értéket, amikor hívunk egy szubrutint vagy függvényt, akkor ennek a változása a kilépés után is megmarad. Ezt hívjuk név szerinti paraméterátadásnak.

Például, írjunk egy szubrutint, ami két értéket megcserél:

```
SUB Swap a, b
    LOCAL t
    t = a
    a = b
    b = t
END SUB
```

Ha valahol ezt meghívjuk:

```
Swap nbr1, nbr2
```

Az eredmény az lesz, hogy az nbr1 és nbr2 értéke felcserélődik.

Ha meg akarjuk tartani a változók eredeti értékeit, akkor ne használjuk ezeket, mint egy általános célú változót egy szubrutin vagy függvény belsejében, mert "rejtélyes" értékváltozást okozhat a szubrutin meghívása után. Ilyen esetben sokkal biztonságosabb helyi változókkal dolgozni.

Tömbök átadása

Egy tömbnek az egyes elemeit, mint egy normális változót szubrutinban vagy függvényben használhatjuk. Például az előbbi Swap szubrutin (lásd fent) helyesen működik:

```
Swap dat(i), dat(i + 1)
```

Ilyen megoldást gyakran használják tömbök rendezésénél.

Teljes tömb is használható egy szubrutin vagy függvény paramétereként, megadva a tömb nevét, amit () követ. Az átadott paraméter típusának (float, integer vagy string) és a benne használt változó típusának azonosnak kell lenni.

A rutinban használt tömb örökölni fogja az átadott tömb méretét, indexelését. Ha szükséges, a tömb mérete akár külön paraméterként is átadható.

A név szerinti tömbátadás azt jelenti, hogy bármilyen változás történik a tömbben egy meghívott alprogramban, az eredetiben is meg fog jelenni. Például, ha a következő program a "Hello World" szöveget nyomtatja ki:

```
DIM MyStr$(5, 5)
MyStr$(4, 4) = "Hello": MyStr$(4, 5) = "World"
Concat MyStr$()
PRINT MyStr$(0, 0)

SUB concat arg$()
    arg$(0,0) = arg$(4, 4) + " " + arg$(4, 5)
END SUB
```

További megjegyzések

Csak egy END SUB vagy END FUNCTION lehet a szubrutinok vagy függvények végén. Ha előbb ki akarunk lépni, használjuk az EXIT SUB vagy EXIT FUNCTION parancsot. Ennek ugyanaz a hatása, mintha szabályosan egy END SUB vagy END FUNCTION paranccsal léptünk volna ki.

Példa egy definiált függvényre

Gyakran van szükség egy speciális parancsra vagy függvényre, amit az MMBasic tartalmaz. Sok esetben ezek felhasználásával magunk is létrehozhatunk egy speciális szubrutint vagy függvényt, amely ezután pontosan úgy fog működni, mint egy beépített parancs vagy függvény.

Például, néha szükség lehet egy karaktersorozat elejét vagy végét csonkító TRIM függvényre. Példaként a következőkben bemutatjuk, hogyan kell ezt megvalósítani.

A függvény első megadandó paramétere az a karaktersor, amelyiket csonkítani kell, a második azt a karaktersort tartalmazza, amit el akarunk távolítani. RTrim\$() a végéről, LTrim\$() az elejéről, Trim\$() mindkét végéről távolít.

```
' trim any characters in c$ from the start and end of s$
Function Trim$(s$, c$)
    Trim$ = RTrim$(LTrim$(s$, c$), c$)
End Function

' trim any characters in c$ from the end of s$
Function RTrim$(s$, c$)
    RTrim$ = s$
    Do While Instr(c$, Right$(RTrim$, 1))
        RTrim$ = Mid$(RTrim$, 1, Len(RTrim$) - 1)
    Loop
End Function

' trim any characters in c$ from the start of s$
Function LTrim$(s$, c$)
    LTrim$ = s$
    Do While Instr(c$, Left$(LTrim$, 1))
        LTrim$ = Mid$(LTrim$, 2)
    Loop
End Function
```

Példák a függvények használatára:

```
S$ = "    ****23.56700  "
PRINT Trim$(s$, " ")
```

Eredmény: "****23.56700"

```
PRINT Trim$(s$, " *0")
```

Eredmény: "23.567"

```
PRINT LTrim$(s$, " *0")
```

Eredmény: "23.56700"

Speciális funkciók és a könyvtár

Számos lehetősége van a gyakorlott Micromite felhasználónak hogy újabb tulajdonságokkal egészítse ki az MMBasic-et, azért hogy speciális műveleteket hajtson végre a rendszer indulásakor, amiket a következőkben ismertetünk.

A legtöbb program nem igényli ezek használatát. Ezeket csak speciális igények esetén használják a gyakorlott felhasználók.

Beágyazott C függvények

Írhatunk C nyelven vagy MIPS assemblerben olyan program modulokat, amelyeket az MMBasic programjainkban felhasználhatunk. Ezeket C függvényeknek (CFunctions), vagy C szubrutinoknak (CSubs) hívjuk, és pontosan úgy használhatók, mint az MMBasic beépített függvényei, vagy rutinjai. Általában ezek a modulok sokkal gyorsabban futnak, mint a BASIC programban szereplő részek, és segítségükkel sokkal könnyebben használható a PIC32 mikrovezérlő speciális hardver tulajdonságai.

A Micromite förmver kiadás tartalmaz egy „Embedded C Modules” nevű könyvtárat, ami egy beágyazott C függvény gyűjtemény és betűkészleteket tartalmaz, és egy hozzátartozó kézikönyvet, amiben megtalálható a rutinok használatának a leírása és útmutatót a saját rutinok, betűkészletek készítéséről.

LIBRARY

A LIBRARY, könyvtár funkció segítségével a felhasználók újabb funkciókat adhatnak az MMBasic programhoz, melyek a nyelv állandó részévé válnak. Például lehetséges, hogy írunk számos speciális szubrutint és függvényt bizonyos bites műveletek elvégzésére. Ezek eltárolhatók, mint egy könyvtár, és részévé válnak MMBasic programnak. A továbbiakban ugyanúgy viselkednek, mint az eredetileg beépített függvények, így ezek immár a nyelv részévé válnak. Egy beágyazott betűkészlet hasonló módon eltárolható, és használható.

Egy összetevő könyvtárba illesztésekor, először meg kell írni és tesztelni a rutinokat úgy, mint minden más normál rutint. Ha ezek megfelelően működnek, akkor használjuk a LIBRARY SAVE parancsot. Ez átírja a rutinokat (annyit, amennyit akarunk) a flash memória egy nem látható részére, ahol elérhető lesz bármelyik BASIC program számára, de nem fognak megjelenni, amikor a LIST parancsot használjuk. Ezen kívül nem kerülnek törlésre, amikor egy új programot töltünk be, vagy a NEW parancsot használjuk. Az elmentett szubrutinok és függvények főprogramból hívhatók vagy akár parancsorból is futtathatók (pontosan úgy, mint egy beépített parancs, vagy függvény).

Néhány megjegyzés:

- A könyvtári rutinok pontosan úgy működnek, mint egy normál BASIC kód, és bármennyi szubrutint, függvényt és CFunction-t tartalmazhatnak. Az egyetlen különbség az, hogy ezek nem láthatók programlistázáskor, és nem törölődnek, amikor egy új program betöltődik.
- A könyvtári rutinok létrehozhatnak és elérhetnek globális változókat, és ugyanazok a szabályok vonatkoznak rájuk mint a főprogramra, például használható az OPTION EXPLICIT parancs is.
- Mikor a rutinok átkerülnek a könyvtár tárterületére, az MMBasic tömöríteni fogja, eltávolítva megjegyzéseket, üres sorokat, és CFunctions hexadecimális kódjait. Ez teszi a könyvtárt nagyon hatékonyvá - különösen nagy fontok betöltésekor. A mentés után a főprogram területe törlődik.
- Használhatjuk a LIBRARY SAVE parancsot többszörösen is. Minden ilyen módon elmentett tartalom az előző könyvtárterülethez hozzáadódik.
- Használhatunk a könyvtárban sorszámozott sorokat, de nem használhatjuk a sorszámokat az egyébként üres sorokra, amivel például egy GOTO utasítás célját kívánjuk megjelölni. Ez azért van, mert a LIBRARY SAVE parancs az üres sorokat eltávolítja.
- A könyvtárban használhatjuk a READ parancsot, de ez az alapértelmezés szerint a program memóriában elhelyezett DATA utasításban szereplő adatokat olvassa. Ha mégis a könyvtárban szereplő DATA adatait akarjuk kiolvasni, akkor az első READ parancs használata előtt a RESTORE parancsot kell kiadni. Ez a mutatót a könyvtár területre állítja.

A LIBRARY DELETE paranccsal lehet a könyvtárból a felhasználói rutinokat törölni. Ez törli a könyvtárat, és ez a terület ismét a normál programok tárolására lesz használható. Az egyetlen másik módja a könyvtár törlésének az MMBasic reszettelése, mely visszaállítja az eredeti konfigurációt, amit a kézikönyv „MMBasic alapállapotba hozása” c. fejezet tartalmaz.

Példaként mentjük el a következőket a könyvtárba:


```
CFunction CPUSpeed
  00000000 3c02bf81 8c45f000 8c43f000 3c02003d 24420900 7ca51400 70a23002
  3c040393 34848700 7c6316c0 00c41021 00621007 3c03029f 24636300 10430005
  00402021 00002821 00801021 03e00008 00a01821 3c0402dc 34846c00 00002821
  00801021 03e00008 00a01821
End CFunction
```

Ez azt eredményezi, hogy egy függvényt (neve: CPUSpeed) hoztunk létre az MMBasic-ben. Ez persze már parancssorból is futtatható:

```
> PRINT CPUSpeed()
40000000
>
```

A könyvtár tartalmának megtekintéséhez használjuk a LIBRARY LIST parancsot. A könyvtár által felhasznált flash memória méretét a MEMORY paranccsal jeleníthetjük meg.

Program inicializálás

A könyvtár tartalmazhat olyan kódot, amely nincs benne egy szubrutinban, függvényben, vagy CFunction-ban. Ez a kód (ha van ilyen) automatikusan lefut a program indulása előtt (azaz, a RUN parancs előtt). Ez a funkció használható állandók inicializálására vagy MMBasic indítási beállítására. Például, ha akarunk néhány állandót definiálni, a könyvtár kódban a következőket kell elhelyezni:

```
CONST TRUE = 1
CONST FALSE = 0
```

Most már a TRUE és FALSE szövegeket hozzáadtuk a nyelvhez, és bármelyik program használhatja.

MM.STARTUP

Szükségünk lehet arra, hogy bizonyos kódot első bekapcsoláskor végrehajtsunk, függetlenül attól, hogy éppen milyen program van a fő memóriában. Ezek például bizonyos hardverinicializálások, opciók beállítása, indítási fejléc kiírása. Ez megvalósítható egy szubrutin létrehozásával, aminek a neve MM.STARTUP és az előbb leírt könyvtárban kell elhelyezni. Amikor a Micromite-ot először bekapcsoljuk, vagy reseteljük, meg fogja keresni ezt a programot, és ha létezik, egyszer lefuttatja.

Például, ha a Micromite-hoz csatlakozik egy RTC elemes óra IC, a könyvtár tartalmazhatja a következő kódot:

```
SUB MM.STARTUP
  RTC GETTIME
END SUB
```

Ennek következtében az RTC időinformációját azonnal átveszi a Micromite minden bekapcsoláskor, vagy reseteléskor.

MM.STARTUP használata hasonló az OPTION AUTORUN funkcióhoz, a különbség az, hogy az AUTORUN opció hatására a teljes program a memóriában kezd futni a kezdetektől, míg az MM.STARTUP csak a szubrutint futtatja. Az AUTORUN opció és az MM.STARTUP együtt is használható. Ebben az esetben a MM.STARTUP szubrutin fut le előbb, majd utána a program a főmemóriában.

Fontos megjegyezni, hogy az MM.STARTUP nem használható az MMBasic általános indítására (tömbök méretének megadására, kommunikációs csatornák megnyitására stb.) amit a program futása előtt szoktunk megtenni. Ennek az az oka, hogy a RUN parancs kiadásakor az MMBasic az értelmező állapotát új indítás szerinti állapotba teszi. Ha mégis ilyen módon szeretnénk az MMBasic-et indítani, akkor ez az indító részt a LIBRARY könyvtárban helyezhetjük el úgy, hogy az interpretert alaphelyzetbe hozó RUN parancs után először ez fusson le, mielőtt a főprogram elindul.

MM.PROMPT

Ha egy szubrutin ezzel a névvel létezik, az MMBasic automatikusan végrehajtja, ahelyett, hogy a parancssor „>” jelét kijelezné. Ez lehetővé teszi, hogy egy általunk megadott kijelentkező karaktersor jelenjen meg a „>” helyett, beállíthatjuk a színeket, definiálhatunk változókat stb. és mindez aktív lesz a prompt megjelenésekor. Ez a szubrutin elhelyezhető a könyvtár területen (ez a javasolt), vagy a programunkban.

Megjegyezzük, hogy az MMBasic törölni fog minden változót, I/O láb beállítást mikor a program elkezd futni, ezért az ebben rutinban történő összes beállítás csak addig érvényes, amíg a parancssoros, azonnali végrehajtási módot használjuk. Változók, állandók, kommunikációk, stb. inicializálása a BASIC programban történjen, amit természetesen a LIBRARY könyvtárban is elhelyezhetünk.

Egy példa a felhasználói prompt megadására:

```
SUB MM.PROMPT
```

```

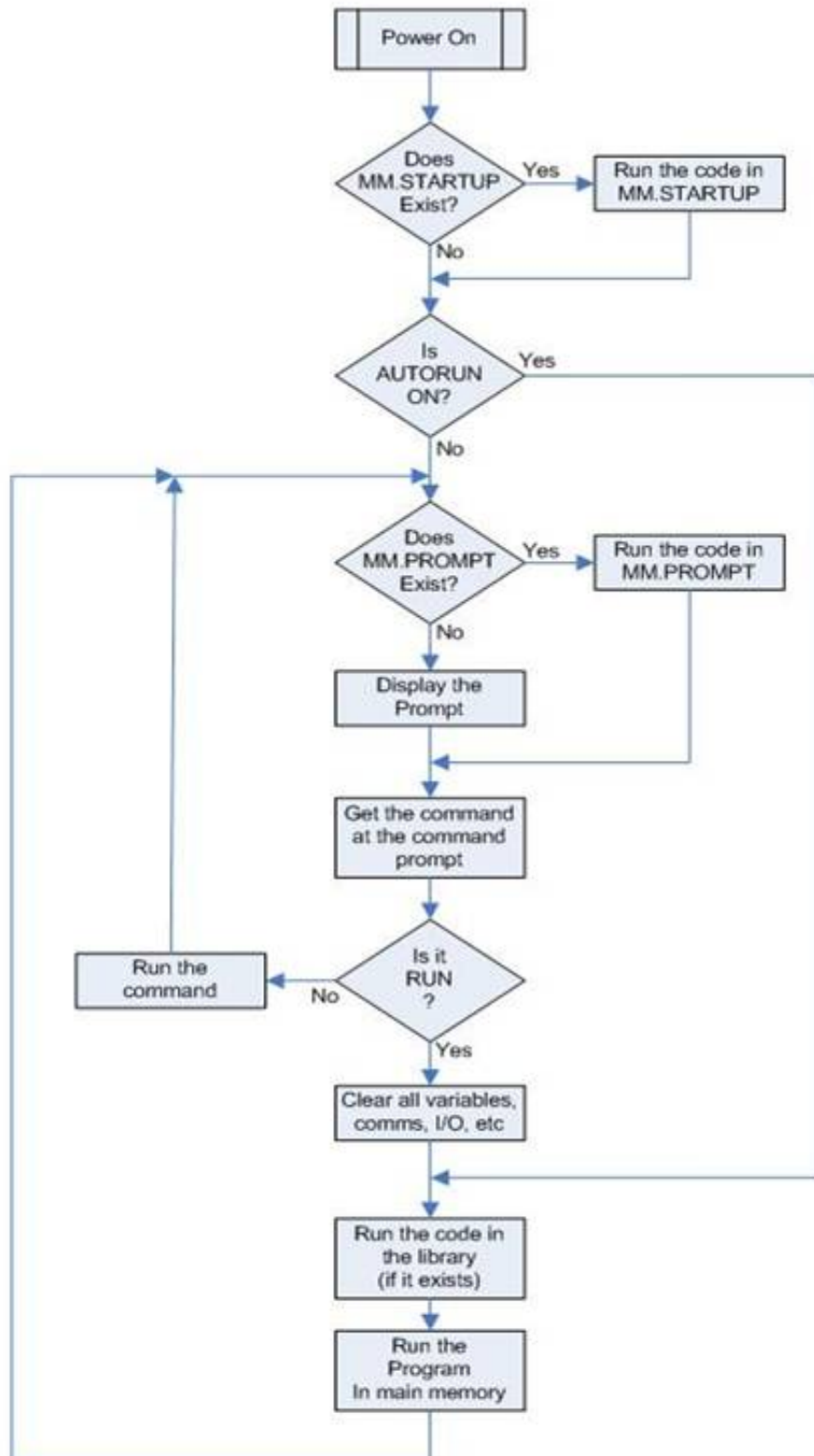
PRINT TIMES "> ";
END SUB

```

Megjegyezzük, hogy a szubrutin belsejében létrehozott állandók nem láthatók a szubrutinon kívül, mert egy szubrutin belsejében definiált állandó lokális a szubrutinban. Azonban a DIM parancs globális változókat hoz létre, emiatt a DIM parancsot kell használni az MM.PROMPT szubrutin belsejében a CONST parancs helyett.

Blokkvázlat

Az MMBasic induláskori működését, és kölcsönhatásait a könyvtárral, és a speciális függvényekkel a legjobban egy blokkvázlatban foglalhatjuk össze. Ez egy magas szintű blokkvázlat (ugyan nem mutatja a CONTINUE parancs okozta komplikációkat), jól szemlélteti az MM.STARTUP és MM.PROMPT szerepét induláskor.



Elektromos jellemzők

Tápellátás

Tápfeszültség tartomány: 2.3 V-tól 3.6V-ig (3.3V névleges). Abszolút maximum 4.0V.

Áramfelvétel 93 mA – 6 mA között, az órajel frekvenciától függ.

Áram alvó állapotban: 40 μ A (plusz az I/O lábak áramfelvétele).

Digitális bemenetek

Logikai alacsony szint: 0-tól 0.65 V-ig

Logikai magas szint: 2.5 V-3.3 V normál lábakon, 2.5 V - 5.5 V az 5 V toleráns lábakon

Bemeneti ellenállás: >1 M Ω . Minden digitális bemeneten Schmitt Trigger van

Kezelhető frekvencia: max. 300kHz (pulzus szélesség 20 nS vagy több) a számláló bemeneteken

Analóg bemenetek

Feszültségtartomány: 0 - 3.3 V

Pontosság: Analóg méréseknél a referencia a tápfeszültség a 28-as és a földpont a 27-es lábon. Ha a tápfeszültség 3.3V, akkor a mérési pontosság tipikusan $\pm 1\%$.

Bemeneti Impedancia: >1 M Ω (pontos mérésnél a forrás impedanciája legyen <5K Ω)

Digitális kimenetek

Tipikus áram kibocsátó vagy nyelő képesség minden I/O lábra: 10 mA

Az előbbieket abszolút maximuma: 15 mA

Maximális áram együttesen az összes I/O lábra: 200 mA

Maximális nyitott kollektor feszültség: 5.5 V

Időzítés pontosság

Minden időzítést használó funkció (időzítő, idő megszakítás, PWM frekvencia, baud rate, stb.) a belső CPU órajeltől függ. A 28 és 44-lábú Micromite gyors RC oszcillátort használ $\pm 0.9\%$ tűréssel, de ez 24°C-on 3.3V tápfeszültségen tipikusan $\pm 0.1\%$ értékű.

PWM kimenet

Frekvencia tartomány: 20 Hz – 500 kHz

Kitöltési tényező: 0% - 100% között 0.1% felbontással 25 kHz alatt

Soros kommunikációs portok

Konzol: Alapértelmezett 38400 baud. Tartomány: 100 bps-230400 bps között (40MHz-nél).

COM portok: Micromite: Alapértelmezett: 9600 baud. Tartomány: 10 bps - 230400 bps között (40 MHz CPU órajelnél). A maximum az órajel sebességtől függ. A maximumot az órajel sebesség korlátozza. Részletesen Ld. Függelék A.

Más kommunikációs portok

SPI 10 Hz – 10 MHz (40 MHz-nél). Korlátozva az órajel negyed frekvenciájára.

I²C 10 kHz – 400 kHz.

1-Wire: Fix 15 kHz.

Flash memória újraírhatósága (endurance)

Legalább 20,000 törlés/írási ciklust garatál a Microchip a PIC32 mikrokontrollereinél.

Minden programmentés egy törlés/írási ciklust jelent. Normál programfejlesztés esetén nagyon valószínűtlen hogy párszáz ciklusnál többet elhasználnunk.

A VAR SAVE paranccsal mentett változók és beállítási lehetőségek (OPTION parancs) szintén a flash memóriába kerülnek, így minden ilyen mentés egy törlési/írási ciklust igényel. A VAR SAVE parancs napi egyszeri használata mellett a várható flash memória élettartama 50 év fölötti.

MMBasic jellemzők

Elnevezési megállapodások

Az MMBasic parancsok, nevek, függvénynevek, címkék, változónevek, stb nem érzékenyek a kis- és nagybetű használatára, így a "Run" és "RUN" azonos, hasonlóan a "doo" és "Doo" változónevekhez.

A változók típusát a DIM paranccsal adhatjuk meg, vagy egy utótagot írva a változónév végére. Például az integer típus esetén az utótag: '%', így a nbr% nevű változó automatikusan integer típusú lesz. A Micromite MMBasic három változótípust támogat:

1. **Lebegőpontos.** A lebegőpontos változó tizedesponttal elválasztott egész, és törtész számokat tárol (pl 45.386). Ezek igen nagy számok is lehetnek, de kisebb pontosságúak, amikor több mint hét számjegyből álló számokat tárolunk, vagy módosítunk. A lebegőpontos változókat megadhatjuk a névhez hozzáírt "!" utótaggal (pl i!, NBR!, stb). Ez az alapértelmezés szerint formátum, akkor is, amikor nem használjuk a "!" utótagot.
2. **64-bites egész.** Az egész típusú változóban pozitív vagy negatív számokat tárolhatnak, akár 19 decimális számjegyből állókat is, a pontosság elvesztése nélkül, viszont törtész nem tartalmazhat. Ezt a formát megadhatjuk a névhez írt "%" utótag megadásával (pl. i%, NBR%, stb). Megjegyezzük, hogy aritmetikai műveletek (különösen összeadás és a szorzás) során nem történik meg az esetleges számábrázolási tartományból való kilépés (túl-vagy alulcsordulás) ellenőrzése. Ilyenkor az eredmény rossz lehet!
3. **Karakterfüzérék.** A karakterfüzér változók karaktersorozatokat tárolnak (pl., "Tom"). A karakterfüzér változók lesznek automatikusan azok, amelyek "\$" karakterrel végződhetnek (pl. name\$, s\$, stb). A karakterfüzér 255 karakter hosszú lehet.

Változónevek és címkék betűvel vagy aláhúzással kezdődhetnek és tartalmazhatnak bármely alfabetikus vagy numerikus karaktert, pontot (.), valamint aláhúzást (_). Hosszúságuk legfeljebb 32 karakter lehet. A változó neve, vagy egy címke nem lehet ugyanaz, mint egy parancs vagy függvény neve. Nem lehetnek az MMBasic nyelv kulcsszavai sem: THEN, ELSE, TO, STEP, FOR, WHILE, UNTIL, MOD, NOT, AND, OR, XOR és AS. Pl. a step = 5 illegális.

Konstansok (állandók)

Szám konstansok kezdődhetnek egy numerikus számjeggyel (0-9) egy decimális állandónál, &H-val egy hexadecimális, &O-val oktális vagy &B-vel egy bináris állandónál. Például &B1000 ugyanaz, mint a 8-as decimális állandó. A &H, &O vagy &B kezdetű értékek mindig 64 bites egész konstansok.

Decimális állandókat megelőzheti a mínusz (-) vagy plusz (+) jel és befejezhető az "E" betűvel, ami után szám jelöli, a hatványkitevő számértékét. Például 1.6E+4 ugyanaz, mint 16000.

Ha az állandó tizedespontot vagy kitevőt tartalmaz, akkor az lebegőpontos konstans jelent. Tizedespont vagy kitevő nélkül mindig 64-bites egész formátumú a konstans.

A szöveg konstansoknál idézőjeleket (" ") kell használni. Pl. "Hello World".

Műveletek és végrehajtási sorrendjük

Az alábbi műveleteket a végrehajtási rangsoruk szerint szerepeltetjük. Azon műveletek, amelyek ugyanazon a szinten helyezkednek el (például a + és -) balról jobbra sorrendben kerülnek feldolgozásra a programsorban. Zárójelek használatával a végrehajtási sorrend módosítható.

Aritmetikai operátorok:

\wedge	hatványozás
$*$ $/$ \backslash MOD	szorzás, osztás, egész osztás modulus (maradék)
$+$ $-$	összeadás, kivonás

Eltolás operátorok:

$x \ll y$ $x \gg y$	A \ll operátor az x 64-bites egész változó bitjeit y bittel balra mozgatja, a nagyobb helyi értékű bitek irányába. Az alacsony helyi értékű bitekre y darab nulla értékű bitet léptet be. A \gg operátor jobbra mozgat.
---------------------	--

Logikai operátorok:

NOT	a jobbra lévő érték logikai inverze
\diamond $<$ $>$ \leq $=<$ \geq $=>$	nem egyenlő, kisebb mint, nagyobb mint, kisebb vagy egyenlő, kisebb vagy egyenlő (alternatív), nagyobb vagy egyenlő, nagyobb vagy egyenlő (alternatív)
=	egyenlő
AND OR XOR	ÉS, VAGY, KIZÁRÓ VAGY

Az AND, OR és XOR bitenkénti műveletet hajtanak végre. Például a PRINT 3 AND 6 művelet kimenete 2 lesz ($011 \& 110 = 010$).

Logikai műveletek akkor eredményezhetnek 0 (nulla) értéket, ha a művelet eredménye hamis, 1 értéket pedig, ha igaz. Például a PRINT $4 > 5$ utasítás kiír egy nulla számot, az $A = 3 > 2$ utasítás hatására A értéke 1 lesz.

A NOT művelet kerül először kiértékelésre (legnagyobb a prioritása). Normál használatnál a tagadandó kifejezést célszerű zárójelezni a helyes kiértékelés érdekében. Például IF NOT (A = 3 OR A = 8) THEN ...

Karakterfüzér műveletek:

+	Két füzér összefűzése
\diamond $<$ $>$ \leq $=<$ \geq $=>$	nem egyenlő, kisebb mint, nagyobb mint, kisebb vagy egyenlő, kisebb vagy egyenlő (alternatív), nagyobb vagy egyenlő, nagyobb vagy egyenlő (alternatív)
=	egyenlő

A füzérek összehasonlítása különbséget tesz a kis és a nagybetűk között. Pl. „A” nagyobb mint „a”

Implementálási (megvalósítási) jellemzők

Maximális programméret (mint sima szöveg): 53KiB. Megjegyezzük, hogy az MMBasic tömöríti (tokenizálja) a programot, mielőtt a flashben eltárolná, így a végső méret változik a szöveg méretétől függően.

A parancssor maximális hossza: 255 karakter.

Egy változónév vagy egy címke maximális hossza: 32 karakter.

Egy tömb maximális dimenziója: 8.

A BASIC parancsok paramétereinek maximális száma: 50

Egymásba ágyazott FOR...NEXT hurkok maximális száma: 10.

Egymásba ágyazott DO...LOOP hurkok maximális száma: 10.

Egymásba ágyazott GOSUB-ok, szubrutinok maximális száma: 50.

Egymásba ágyazott többsoros IF...ELSE...ENDIF utasítások maximális száma: 10.

Felhasználó által definiált szubrutinok és függvények maximális száma (együttesen): 100 (200 MMplus esetén)

A megszakítás bemenetként konfigurálható lábak száma: 10

Az egyszeres pontosságú lebegőpontos számok ábrázolási tartománya:

Minimum 1.17549435e-38. és Maximum 3.40282347e+38

A 64-bites előjeles egészek ábrázolási tartománya: ± 9223372036854775807 .

Maximális karakterfüzér hossz: 255 karakter.

A BASIC program sorainak a száma: 65000.

A PULSE parancssal háttérben kiadható független pulzusok száma: 5.

Kompatibilitás

Az MMBasic a Microsoft's GW-BASIC implementációja. Számos kisebb különbség van, köszönhetően a fizikai és gyakorlati megfontolásoknak, de a legtöbb szabványos BASIC parancs és függvény azonos. A GW-BASIC egy online kézikönyve elérhető a <http://www.antonis.de/qbebooks/gwbasman/index.html> linken, és igen részletes leírást ad a parancsokról, és a függvényekről.

MMBasic tartalmaz még számos modern programozási struktúrát, amiket az ANSI Standard for Full BASIC (X3.113-1987) vagy az ISO/IEC 10279:1991 szabványok definiálnak.

Tartalmazza a SUB/END SUB, DO WHILE ... LOOP, SELECT...CASE utasításokat, és a strukturált IF .. THEN ... ELSE ... ENDIF utasításokat.

Előre megadott, csak olvasható változók

Ezeket a változókat az MMBasic állítja, és nem változtatható egy futó programban.

MM.VER	A főmver verziószáma lebegőpontos számként ábrázolva, aa.bbccc formában ahol aa a fő verziószám, bb az alverziószám és cc a revízió száma. Például az 5.03.00 verzió esetén 5.3 alakú a megjelenítés, míg 5.13.01 esetén ez 5.0301
MM.DEVICES\$	Egy karakterfüzér, ami azt jelzi, hogy melyik az az eszköz vagy platform, amelyen az MMBasic fut. Jelenleg ez a változó alábbiak egyike: "Maximite" a szabványos Maximite és kompatibilis verziói. "Colour Maximite" a colour Maximite-on vagy UBW32-n fut. "DuinoMite", DuinoMite családon fut. "DOS", Windows DOS-ablakában fut. "Generic PIC32" általános változata a PIC32-n futó MMBasic-nek. "Micromite" a PIC32MX150/250-n fut "Micromite MkII" a PIC32MX170/270-n fut "Micromite Plus" a PIC32MX470-n fut "Micromite Extreme" a PIC32MZ sorozaton fut
MM.ERRNO MM.ERRMSG\$	Ha egy utasítás hibát okozott, amit figyelmen kívül akarunk hagyni, ezeket a változókat egy nem nulla értékkel és egy hibaüzenet szövegével kell feltölteni, ami majd a konzolon fog megjelenni. A RUN, az ON ERROR IGNORE vagy az ON ERROR SKIP parancsok nullázzák a változót, és üres füzért töltenek a karakteres változóba.
MM.HRES MM.VRES	Az LCD kijelző panel horizontális és vertikális felbontása egész számként megadva, pixelben. (Ha konfiguráljuk)
MM.FONTHEIGHT MM.FONTWIDTH	Az aktuális betűkészlet karaktereinek a magassága és a szélessége egész számként megadva, pixelekben.
MM.WATCHDOG	Egy egész szám, ami igaz, ha az MMBasic újraindult a Watchdog túlszordulás miatt (ld. WATCHDOG parancsot). Hamis, ha az MMBasic normálisan indult.
MM.I2C	I2C írási vagy olvasási parancsot követően ez az egész változó jelzi a művelet eredményét a következők szerint: 0 = A parancs hiba nélkül befejeződött 1 = NACK válasz vétele 2 = Parancs időtúllépés
MM.ONEWIRE	Az egy vezetékes reszet funkciót követően, ez az egész típusú változó jelzi a művelet eredményét: 0 = Eszköz nem található 1 = Eszköz megtalálva

Parancsok

Az I²C, egyvezetékes és az SPI kommunikációval kapcsolatos funkciók nem szerepelnek ebben a fejezetben, a részletes leírásuk megtalálható a dokumentum végén lévő „A”, „B”, „C” és „D” függelékekben.

Szögletes zárójelek jelzik, hogy a paraméter, vagy a karakterek megadása nem kötelező, opcionális.

` (egyszeres idézőjel)	Kezdődik egy megjegyzés, és bármilyen ezt követő szöveget figyelmen kívül hagy. Megjegyzések bárhol elhelyezhetők a sorban.
? (kérdőjel)	PRINT parancs röviden.
AUTOSAVE	Automatikus programbeviteli módba lépés. Ez a parancs fogadja a sorokból álló szöveget a konzol soros bemenetén és elmenti a memóriába. Ez az üzemmód befejezhető a Control-Z megadásával, és a bevitt program a memóriába kerül, felülírva annak előző tartalmát. A parancs bármikor megszakítható a Control-C kombinációval, és a programmemória érintetlen marad. Ez az egyik mód egy BASIC program Micromite-ba írásába. A terminálemulátoron át küldött programot az MMBasic fogadja, és eltárolja a programmemóriába. Arra is használható, hogy egy kis programot közvetlenül a konzol bemenetéről vigyünk be.
BOX x1, y1, w, h [,lw] [,c] [,fill]	Egy dobozt rajzol x1, y1 pontról w pixel szélesen h pixel magasan a csatlakoztatott LCD panelen. 'lw' a doboz vonalainak a szélessége, 0 is lehet, alapértelmezett értéke: 1. 'c' a vonalak színe, ha nem adjuk meg, akkor az alapértelmezés szerinti előtérszín. 'fill' a dobozt kitöltő szín. Elhagyható, vagy -1-re állítva a doboz nem lesz színnel kitöltve. Lásd az "Rajzoló parancsok" fejezetet a színek megadásához, és a grafikus koordináták értelmezéséhez.
CIRCLE x, y, r [,lw] [,a] [,c] [, fill]	Kör rajzolása x,y középponttal, r sugárral az LCD panelen. 'lw' a vonal szélessége, 0 is lehet, alapértelmezett értéke:1. 'a' a torzítási arány egy lebegőpontos szám, alapértelmezett értéke:1.0, ami szabályos kört eredményez. Például, ha értéke: 0.5 akkor egy oválist rajzol, aminek szélessége fele a magasságának. 'c' a körvonal színe, ha nem adjuk meg, akkor az alapértelmezés szerinti előtérszín. 'fill' a kört kitöltő szín. Elhagyható, vagy -1-re állítva a kör nem lesz színnel kitöltve Lásd az "Rajzoló parancsok" fejezetet a színek megadásához, és a grafikus koordináták értelmezéséhez.
CLS [colour]	Az LCD kijelzőt a colour-al megadott színre törli. Ha ezt nem adjuk meg, akkor az aktuális alapértelmezés szerinti háttérszínt használjuk.
COLOUR fore [, back] vagy COLOR fore [, back]	Beállítja az alapértelmezés szerinti színeket a színeket használó LCD panelre író parancsoknál. 'fore' az előtér színe, 'back' a háttér színe. Ez utóbbi opcionális, ha nem adjuk meg akkor fekete lesz.
CSUB name(type[,type]) rtype hex [[hex[...] hex [[hex[...] END CSUB vagy	Definiál egy bináris kódrészletet, egy beágyazott programmodult a programunkban, ami C-ben vagy MIPS gépi kódban lett megírva. Ez a modul úgy jelenik meg az MMBasic programban, mint egy adott nevű parancs, vagy függvény ("name"), és ugyanúgy használható, mint egy beépített függvény, vagy parancs. A célja bizonyos kódrészletek jóval gyorsabb futtatása.

<p>CFUNCTION name type ,type]</p> <p>hex [[hex[...]</p> <p>hex [[hex[...]</p> <p>END CFUNCTION</p>	<p>Ez a parancs tartalmazza a modul kódját hexadecimális formában. Ezt az MMBasic automatikusan elhelyezi a programban, mikor azt elmenti. Minden "hex" résznek pontosan nyolc hexa számjegyből kell állnia, mert ez kódolja a memória egy 32-bites szavát.</p> <ul style="list-style-type: none"> • Minden "hexa" szót szét kell elválasztani egy vagy több szóközzel és többsoros "hex" szavakat kell használni. A parancsot le kell zárni egy megfelelő END CSUB vagy END CFUNCTION parancssal. • Az első "hexa" szó lesz az eltolás (32 bites szóméretben), ami megadja a belépési pontját a beágyazott rutinnak. Általában a main () függvényre mutat. • Több beágyazott rutin használható a programban különböző modulok és hozzá tartozó nevük megadásával. (Egy másik modul másik "name"). • Programfuttatáskor az MMBasic kihagyja a programsorokban megjelenő CSUB vagy CFUNCTION parancsokat, így azok bárhol elhelyezhetők a programban. <p>Bármilyen hiba az adatok formátumában a program mentésekor hibaüzenetet generál.</p> <p>Minden paraméter típusa megadható a definícióban. Például:</p> <pre>CSub MySub integer, integer, string</pre> <p>Ez azt adja meg, hogy három paraméterünk lesz, az első kettő integer, a harmadik egy karakterfüzér.</p> <p>Hasonlóan a CFunction bemenő paramétereinek a definiálásához, megadhatjuk a visszatérési érték típusát is. Például a következő egy karakterfüzérrel tér vissza:</p> <pre>CFunction MyFunct(integer, string) string</pre> <p>A 'name' névvel létrehozott parancsot normál parancsként, vagy függvényként lehet a programban használni, néhány kikötéssel:</p> <ul style="list-style-type: none"> • Akár tíz érték adható meg a paraméterként ("arg1 ',' arg2", stb), és ezek átadódnak a beágyazott C rutinnak, mint mutatók, amelyek a paraméterekre mutatnak. (azaz, az eredmény a kifejezés). • Ha egy változó vagy tömb szerepel egy C rutin argumentumaként, akkor kap egy memóriára mutatót és a C rutin megváltoztathatja ezt a memóriát, aminek tartalmát visszaadja a hívónak. • Tömbök esetén, azokat üres zárójelek között kell átadni pl. az Arg (). A C függvényben az argumentum, mint mutató a tömb első elemére mutat. <p>További részletek a „Embedded C Modules” című dokumentációban található a Micromite förmvert tartalmazó fájlban.</p>
<p>CLEAR</p>	<p>Törli az összes változót és felszabadítja a memóriát.</p> <p>Ld. ERASE parancsot a tömbváltozók törléséhez.</p>
<p>CLOSE [#]nbr [, [#]nbr] ...</p>	<p>Bezárja a soros kommunikációs portot, amit előtte megnyitottunk a 'nbr' fájlzámmal. # használata opcionális. Ld. még az OPEN parancsot.</p>
<p>CONST id = expression [, id = expression] ... etc</p>	<p>Létrehoz egy állandót, egy konstanst, amelyet már nem lehet ezután megváltoztatni.</p> <p>"id" az az azonosító, amely ugyanazokat a szabályokat követi, mint a változóknál. Az azonosítónak lehet típus utótagja (!, %, Vagy \$), de ez nem szükséges. Ha megadjuk, akkor meg kell egyeznie a "expression" típusával.</p> <p>'expression' az azonosító értéke. Ez lehet egy normál kifejezés is (beleértve a felhasználó által definiált funkciókat), ami az állandó létrehozásakor kapja meg az értékét.</p> <p>Ha az állandót a szubrutinon vagy függvényen kívül definiáltunk globális lesz és az egész programban látható. Ha az állandót egy szubrutin vagy</p>

	függvény belsejében, lokálisan hozunk létre, akkor a nevével azonos globális konstans vagy változó értékét a szubrutinban átmenetileg felülírja.
CONTINUE	<p>Folytatja a program futását, ami megszakadt egy END utasítás, hiba, vagy a CTRL-C hatására. A program továbbindul a megállási pontot követő utasítástól.</p> <p>Ne feledje, hogy a program helyreállítása nem mindig lehetséges - ez különösen a grafikus, a beágyazott hurkok és / vagy a beágyazott szubrutinok és funkciók komplex programjaira vonatkozik.</p>
CONTINUE DO vagy CONTINUE FOR	Elugrik a DO/LOOP vagy a FOR/NEXT hurok végére. A hurokban maradás feltétele itt tesztelve lesz, és ha érvényes, akkor folytatja a ciklust.
CPU speed	<p>A processzor órajel-frekvenciáját lehet vele előállítani.</p> <p>'speed': MHz-ben kell megadni. Értéke MM28/MM44 esetén 48, 40, 30, 20, 10, vagy 5 lehet.</p> <p>MM+ esetén ezek 120, 100, 80, 60, 48, 40, 30, 20, 10 vagy 5 lehet.</p> <p>A Micromite által felvett áram arányos az órajellel, így felére csökkentett órajelnél az áram is kb. felére csökken.</p> <p>A CPU bekapcsolásakor az alapértelmezett frekvencia MM28/MM44 esetén 40 MHz, MM+ esetén pedig 100 MHz..</p> <p>Ha a sebesség megváltozik, az MMBasic minden időzítés funkciót automatikusan korrigál, hogy azok ne változzanak, és a konzol adatátviteli sebessége is változatlan marad. A soros kommunikációs portok nyitva maradnak a sebesség változtatás alatt és a sebesség ennek megfelelően módosul. Az órajel változtatásakor, míg a sebesség megváltozik, néhány karakter elveszhet vagy sérülhet. A megnyitott SPI, I²C és PWM funkciók is változni fognak az órajel változás miatt, ezért célszerű az órajel változtatása előtt ezeket bezárni, és a sebesség változtatás után újranyitni.</p>
CPU SLEEP vagy CPU SLEEP seconds [,abortpin]	<p>A CPU alvó módba kerül. Ebben az üzemmódban a futó program leáll és vele együtt az az áramfelvétel jelentősen lecsökkenhet, akár 40 µA-re. Kétféle SLEEP utasításforma van</p> <p>Az első formánál az ébresztést a WAKEUP láb végzi (ld. a lábkiosztást a kézikönyv elején). A parancs hatására automatikusan digitális bemenetnek konfigurálja a program ezt a lábat, és annak bármely állapotváltása (pl. magas-alacsony vagy alacsony-magas átmenet) felébreszti a CPU-t. Az ébredéshez szükséges idő ilyenkor kevesebb mint 1 msec.</p> <p>Az IR parancs osztozik a WAKEUP lábbal, emiatt ha az IR szenzor aktív, a CPU felébred a távvezérlő gomb megnyomására és az MMBasic azonnal dekódolja a jelet, és végrehajtja az IR bemenethez tartozó megszakítást.</p> <p>A második formában (CPU SLEEP seconds), a parancs SLEEP-be küldi a CPU-t a megadott "seconds" időre (a pontosság ± 20%). A TIMER funkció és a belső óra/naptár az alvás befejezésekor frissül.</p> <p>Az időzített alvás korábban befejezhető, ha "abortpin" meg van megadva. Ez lehet bármely I/O láb, és bármilyen állapotváltozás (pl. magasról alacsonyra vagy alacsonyról magasra) megszakítja az alvást. Egy másodpercig is tarthat az abort jel felismerése, így az állapotváltozást legalább erre az időre fenn kell tartani.</p> <p>Megjegyzések:</p> <ul style="list-style-type: none"> • A CPU a SLEEP parancs közepén alszik el, és amikor felébred, onnan folytatja a normál programvégrehajtást. • Alvás alatt minden kommunikáció (soros, SPI, I2C és 1-Wire) és PWM befagy. Mikor a CPU kijön az alvásból, akkor folytatja a programfutást. Javasolt, hogy zárjuk le ezeket a funkciókat az alvásba történő belépés előtt, mivel növelhetik a tok SLEEP alatti áramfelvételét.

	<ul style="list-style-type: none"> • Külső áramkörök és a programfunkciók extra áramot jelenthet az alvás alatt.. Lásd a „<i>Micromite speciális funkciók</i>” részt azokhoz a lépésekhez, amelyeket az alvás alatti áram minimalizálása érdekében kell végrehajtani. • Minden időzítő funkció befagy alvás alatt, ez vonatkozik a belső óra/naptárra és a háttérben működő pulzus parancsokra is. • A konzolon kiadott CTRL-C nem hozza ki az alvásból a tokot.
CPU RESTART	<p>Újraindítja a processzort.</p> <p>Ez minden változót törölni fog, és mindent reszetel, azaz alapállapotba állít (pl. időzítők, COM portok, I²C, stb.). Hasonló a bekapcsoláshoz, annak bejelentkező üzenete nélkül.</p> <p>Ha az OPTION AUTORUN opció be lett korábban állítva, a memóriában lévő program újraindul.</p>
DATA constant[,constant]...	<p>Numerikus és fűzér állandók érhetők el a READ paranccsal.</p> <p>Általában a karakterlánc konstansokat idézőjelek között kell elhelyezni (" "). Ez alól kivétel, ha a fűzér csak alfanumerikus karakterekből áll, amelyek nem jelentenek MMBasic kulcsszavakat (pl. THEN, WHILE, stb.) Ebben az esetben idézőjelek nem szükségesek.</p> <p>Numerikus konstansok lehetnek kifejezések is, mint 5*60.</p>
DATES = "DD-MM-YY" vagy DATES = "DD/MM/YY"	<p>Beállítja a belső óra/naptár dátumát.</p> <p>DD, MM and YY számok, pl.: DATES = "28-7-2014"</p> <p>Bekapcsoláskor a dátum: "01-01-2000" lesz.</p>
DEFINEFONT #Nbr hex [[hex[...] hex [[hex[...] END DEFINEFONT	<p>Egy beágyazott betűkészletet definiál, amely együtt használható a beépített betűkészletekkel, vagy ezekkel kicserélhető és így használhatjuk az LCD panelen, Pontosan úgy működnek, mint a beépített fontok. (pl. kiválaszthatók a FONT paranccsal, vagy megadhatjuk a TEXT parancsban.</p> <p>'#Nbr' a betűkészlet (font) azonosító száma 1-16 között. Ez a szám megegyezhet egy beépített készlet azonosítószámával, de ilyenkor a beépített betűkészletet lecseréli.</p> <p>Minden 'hex' pontosan nyolc hexa számjegyből áll, és betűközökkel, vagy új sor karakterekkel kell ezeket szétválasztani.</p> <ul style="list-style-type: none"> • Több sorból álló 'hex' szavak használhatók, az utolsó sor után kell szerepeltetni az END DEFINEFONT parancsot. • Egy programban több különböző azonosító számmal megadott beágyazott betűkészlet használható, amelyik különböző típusokat definiál. • Az MMBasic a programfutás alatt átugorja mindegyik DEFINEFONT parancsot, így ezek a program bármelyik részén elhelyezhetők. <p>Az adatformátumban szereplő esetleges hibákat a program mentésekor az MMBasic jelzi.</p>
DIM [type] decl [,decl]... ahol 'decl': var [length] [type] [init] 'var' a változó neve, opcionális dimenzióval 'length' használt a fűzér maximális hosszának a megadására 'n' mint LENGTH n 'type' AS FLOAT, AS INTEGER, AS STRING egyike 'init' az érték, amivel inicializáljuk a változót és formája: = <expression>	<p>Egy vagy több változót deklarál (azaz létrehoz változó nevet és jellemzőit az interpreter számára).</p> <p>Ha OPTION EXPLICIT parancsot használjuk (ami javasolt) a DIM és a LOCAL parancsok használata az egyetlen lehetőség változók létrehozására. Ha ezt nem használjuk, akkor a DIM parancs opcionális, és ha nem használjuk, a létrehozott változót az első rá történő hivatkozásnál hozzuk létre.</p> <p>A változó típusa (string, float vagy integer) három módon adható meg:</p> <ol style="list-style-type: none"> 1. Típus utótag használata (!, % vagy \$ lebefor float, integer or string). Például: DIM nbr%, amount!, name\$ 2. A FLOAT, INTEGER vagy STRING kulcsszavak használata, közvetlenül a DIM után, és mielőtt a változókat felsorolnánk. A megadott típus ezután érvényes lesz a felsorolt változókra. Például:

<p>Egy változónál egy kifejezést használunk, tömböknél vesszővel elválasztott kifejezéseket zárójelekkel közrefogva.</p> <p>Példák</p> <pre> DIM nbr(50) DIM INTEGER nbr(50) DIM name AS STRING DIM a, b\$, nbr(100), str\$(20) DIM a(5,5,5), b(1000) DIM str\$(200) LENGTH 20 DIM STRING str(200) LENGTH 20 DIM a = 1234, b = 345 DIM STRING str = "text" DIM x%(3) = (11, 22, 33, 44) </pre>	<pre> DIM AS STRING first_name, last_name, city </pre> <p>3. Microsoft konvenció használata az: "AS" után kell megadni FLOAT, INTEGER vagy STRING kulcsszavak valamelyikét minden változó után. Ha ezt a módszert használjuk a típust minden változó előtt meg kell adni. Például:</p> <pre> DIM amount AS FLOAT, name AS STRING </pre> <p>A lebegőpontos és egész változók kezdeti értéke nulla lesz a deklaráció után, a füzérek kezdeti értéke üres fűzér ("") lesz. A változónak a deklaráláskor adhatunk más kezdőértéket, az egyenlőségjel (=) után megadva egy kifejezést vagy értéket a definíció után. Például:</p> <pre> DIM AS STRING city = "Perth", house = "Brick" </pre> <p>Ha kifejezést szerepeltetünk (akár más változókkal) ez a DIM parancs kiadásakor történik meg a kiértékelése. Több példa szerepel még "Változók definiálása és használata" című részben.</p> <p>Ahogy deklaráltunk egyszerű változókat a DIM paranccsal, hasonló módon deklarálhatunk tömbös változókat (indexelt változókat, tetszőleges dimenzióval). A változó neve után kell szerepeltetni a tömb dimenzióját, egy számokból álló vesszővel elválasztott listával, amit zárójelbe rakunk. Például:</p> <pre> DIM array(10, 20) </pre> <p>Minden szám az adott dimenzióhoz tartozó elemek számát jelöli. Alapesetben a számok 0-tól kezdődnek, de az OPTION BASE paranccsal ezt a kezdőértéket 1-re változtathatjuk.</p> <p>A fenti példa kétdimenziós tömböt specifikál, az első dimenzióban 11 elemes (0-10), a második dimenzió mérete 21 lesz (0-20). Az összes tömbelem száma $11 \cdot 21 = 231$, és mivel minden lebegőpontos elem 4 bájtot foglal el, a memóriában ez $231 \cdot 4 = 924$ bájton helyezkedik el. (Ha a típus egész, akkor annak helyfoglalása 8 bájt elemenként).</p> <p>A füzérek alapesetben 255 bájtot (karaktert) +1 bájtot foglalnak el a memóriában, és ezért gyorsan megtölthetik a memóriát. Minden fűzér végén a +1 bájt tartalmazza a fűzér tényleges hosszát. Célszerű ezért a LENGTH kulcsszót használni, amivel megadhatjuk a fűzér által elfoglalt hely hosszát karakterben ('n'). Ez az érték 1-255 között lehet.</p> <p>Például: DIM str\$(5, 10) egy fűzértömböt deklarál, 66 elemmel és ennek elfoglalása $6 \cdot 11 \cdot 256 = 16.896$ bájt. Ellenben a:</p> <pre> DIM AS STRING str (5, 10) LENGTH 20 </pre> <p>Csak $6 \cdot 11 \cdot 21 = 1.386$ bájtot foglal el.</p> <p>Ha a fűzér, amit tárolni akarunk a változóban nagyobb, mint 'n' egy hiba-üzenet jelenik meg. A LENGTH kulcsszóval létrehozott fűzértömbök ugyanúgy működnek, mint más fűzértömbök. Nem tömbös füzérek definiálásánál is használhatjuk a LENGTH kulcsszót.</p> <p>A fenti példánál használhatja a Microsoft szintaxisát, a típus és a hossz megadásánál. Például:</p> <pre> DIM str (5, 10) LENGTH 20 AS STRING </pre> <p>A tömbök ugyanúgy inicializálhatók deklarációkor kezdőértékekkel, az egyenlő jel, és zárójelbe zárt vesszővel elválasztott lista segítségével. Például:</p> <pre> DIM INTEGER nbr(4) = (22, 44, 55, 66, 88) </pre> <p>vagy:</p> <pre> DIM str\$(3) = ("foo", "boo", "doo", "zoo") </pre> <p>Megjegyezzük, hogy az inicializáló értékek számának, meg kell egyeznie a tömb elemeinek a számával, ha OPTION BASE értékét 1-re állítottuk. Többdimenziós tömbök esetén először az első dimenziót töltjük fel, utána a másodikat stb.</p>
<p>DO <statements> LOOP</p>	<p>Örökké futó programhurok; az EXIT parancs használható a hurokból való kilépésre, vagy a vezérlést át lehet irányítani a hurkon kívülre a GOTO vagy RETURN (ha szubrutinban vagyunk) utasítással.</p>

DO WHILE expression <statements> LOOP	Hurokban marad, míg az "expression" igaz (ugyanaz, mint a régi WHILE-WEND hurok, ami szintén szerepel az MMBasic-ben). Ha induláskor az "expression" hamis a hurok egyszer sem lesz végrehajtva ("előletesztelő ciklus").																												
DO <statements> LOOP UNTIL expression	Hurokban marad, az UNTIL-t követő "expression" igaz. Mivel a tesztelés a hurok végén van, és ha induláskor az "expression" hamis, a hurok akkor is egyszer végrehajtásra kerül ("hátulatesztelő ciklus").																												
EDIT	<p>Elindítja a teljes képernyős szerkesztőt.</p> <p>Minden szerkesztő billentyű működik a VT100 terminál emulátorral, így a szerkesztés végrehajtható konzol soros vonalán keresztül. Az editor tesztelve let a Tera Term és PUTTY programokkal a Windows PC-n.</p> <p>Belépéskor a kurzor automatikusan a legutóbb szerkesztett sorra áll, illetve, ha volt egy hiba, a program futása közben, akkor arra a sorra, ami a hibát okozta.</p> <p>A szerkesztő billentyűk:</p> <table> <tr> <td>Bal/Jobb nyilak</td><td>A kurzort mozgatja a soron belül.</td></tr> <tr> <td>Fel/Le nyilak</td><td>A kurzort mozgatja egy sorral felfelé, vagy lefelé</td></tr> <tr> <td>Page Up/Down</td><td>A program egy lapjával felfelé vagy lefelé mozog.</td></tr> <tr> <td>Home/End</td><td>A kurzort mozgatja a sor elejére vagy végére. A második Home/End a program elejére vagy a végére áll.</td></tr> <tr> <td>Delete</td><td>Törli a karaktert, ahol a kurzor áll. Ha ez egy sorváltó karakter, akkor a két sort egyesíti.</td></tr> <tr> <td>Backspace</td><td>A kurzor előtti karaktert törli.</td></tr> <tr> <td>Insert</td><td>Átkapcsol a beszúró vagy felülíró üzemmód között.</td></tr> <tr> <td>Escape Key</td><td>Bezárja a szerkesztőt mentés nélkül (előtte megerősítést kér).</td></tr> <tr> <td>F1</td><td>Menti a szerkesztett szöveget és kilép.</td></tr> <tr> <td>F2</td><td>Ment, kilép, és futtatja a programot.</td></tr> <tr> <td>F3</td><td>Elindítja a kereső tevékenységet.</td></tr> <tr> <td>SHIFT F3</td><td>Újra keres használva az F3 gomb után bevitt szöveget.</td></tr> <tr> <td>F4</td><td>Szöveget jelöl meg kivágáshoz és másoláshoz (ld. lent).</td></tr> <tr> <td>F5</td><td>Az előzőleg kivágott vagy másolt szöveget beilleszti.</td></tr> </table> <p>Mikor a szövegmegjelölő módban vagyunk (belépés F4-el) a szerkesztő lehetővé teszi, hogy a nyílbillentyűkkel a szöveget kijelöljünk, ami törölhető, a vágólapra kivágható, vagy egyszerűen oda másolható. Az alsó sorban a státuszsor megváltozik, mutatva a funkcióbillentyűk új funkcióit.</p> <p>A szerkesztő képes hosszabb sorokat is kezelni, mint ami a képernyőn látszik, de karakterek képernyő szélé után már nem láthatók. Ilyenkor egy új sor karakter beiktatásával (Enter) érdemes a sort kettétörni, és persze később, ha szükséges, ez a sortörés megszüntethető a két sor egyesítésével.</p>	Bal/Jobb nyilak	A kurzort mozgatja a soron belül.	Fel/Le nyilak	A kurzort mozgatja egy sorral felfelé, vagy lefelé	Page Up/Down	A program egy lapjával felfelé vagy lefelé mozog.	Home/End	A kurzort mozgatja a sor elejére vagy végére. A második Home/End a program elejére vagy a végére áll.	Delete	Törli a karaktert, ahol a kurzor áll. Ha ez egy sorváltó karakter, akkor a két sort egyesíti.	Backspace	A kurzor előtti karaktert törli.	Insert	Átkapcsol a beszúró vagy felülíró üzemmód között.	Escape Key	Bezárja a szerkesztőt mentés nélkül (előtte megerősítést kér).	F1	Menti a szerkesztett szöveget és kilép.	F2	Ment, kilép, és futtatja a programot.	F3	Elindítja a kereső tevékenységet.	SHIFT F3	Újra keres használva az F3 gomb után bevitt szöveget.	F4	Szöveget jelöl meg kivágáshoz és másoláshoz (ld. lent).	F5	Az előzőleg kivágott vagy másolt szöveget beilleszti.
Bal/Jobb nyilak	A kurzort mozgatja a soron belül.																												
Fel/Le nyilak	A kurzort mozgatja egy sorral felfelé, vagy lefelé																												
Page Up/Down	A program egy lapjával felfelé vagy lefelé mozog.																												
Home/End	A kurzort mozgatja a sor elejére vagy végére. A második Home/End a program elejére vagy a végére áll.																												
Delete	Törli a karaktert, ahol a kurzor áll. Ha ez egy sorváltó karakter, akkor a két sort egyesíti.																												
Backspace	A kurzor előtti karaktert törli.																												
Insert	Átkapcsol a beszúró vagy felülíró üzemmód között.																												
Escape Key	Bezárja a szerkesztőt mentés nélkül (előtte megerősítést kér).																												
F1	Menti a szerkesztett szöveget és kilép.																												
F2	Ment, kilép, és futtatja a programot.																												
F3	Elindítja a kereső tevékenységet.																												
SHIFT F3	Újra keres használva az F3 gomb után bevitt szöveget.																												
F4	Szöveget jelöl meg kivágáshoz és másoláshoz (ld. lent).																												
F5	Az előzőleg kivágott vagy másolt szöveget beilleszti.																												
ELSE	Bevezet egy alapértelmezett feltételt egy többsoros IF utasításban. Lásd a többsoros IF utasítás leírását a részletekért.																												
ELSEIF expression THEN vagy ELSE IF expression THEN	Bevezet egy második feltételt egy többsoros IF utasításban. Lásd a többsoros IF utasítás leírását a részletekért.																												
ENDIF vagy END IF	Többsoros IF utasítás lezárása. Lásd a többsoros IF utasítás leírását a részletekért.																												

END	A futó program befejezése után visszatér a parancssorba.
END FUNCTION	Felhasználó által definiált függvény végét jelöli. Lásd a FUNCTION parancsot. Minden függvénynek kötelezően egyetlen hozzá tartozó END FUNCTION utasítása van. Használjuk az EXIT FUNCTION utasítást, ha a függvény belsejéből akarunk valahol kilépni.
END SUB	Felhasználó által definiált szubrutin végének a jelzése. Ld. a SUB parancsot. Minden szubrutinnak kötelezően egyetlen END SUB utasítása van. Használjuk az EXIT SUB utasítást, ha egy rutin belsejéből akarunk kilépni.
ERASE variable [,variable]...	Tömbváltozókat törli és memóriát szabadít fel. A CLEAR parancsot használja az összes változó törlésére, a tömbváltozókkal együtt.
ERROR [error_msg\$]	Hibát okoz, és kilépünk a programból. Ezt rendszerint a hibakereséskor használjuk.
EXIT DO EXIT FOR EXIT FUNCTION EXIT SUB	EXIT DO korai kilépést biztosít a DO...LOOP hurokból. EXIT FOR DO korai kilépést biztosít a FOR...NEXT hurokból. EXIT FUNCTION DO korai kilépést biztosít a definiált függvényből. EXIT SUB korai kilépést biztosít a definiált rutinból. A régebbi szabványú EXIT (kilépés a DO hurokból) is támogatott.
FONT [#]font-number, scaling	Beállítja az alapértelmezett betűkészletet a grafikus LCD panelnél. A betűkészletekre számokkal hivatkozunk. Például #2 (a „#” használata opcionális). A használható betűkészleteket a „Grafikus LCD panel használata” fejezet betűkészlet alfejezete részletesen ismerteti. A 'scaling' értéke 1-től 15-ig változhat és a karakterek méretét sokszorozza meg. Például a 2 megduplázza a karakter magasságát és szélességét.
FOR counter = start TO finish [STEP increment]	Egy FOR-NEXT hurkot hoz létre. 'counter' kezdet értéke 'start' és a counter az 'increment' értékével változik (alapértelmezés: 1), amíg 'counter' egyenlő nem lesz 'finish'-el. Az 'increment' lehet egész vagy lebegőpontos szám. Felhívjuk a figyelmet arra, hogy az "increment" nevű lebegőpontos törtszám használata kerekítési hibákat halmozhat fel a "számlálóban", ami a hurok korai vagy késői befejezéséhez vezethet. 'increment' lehet negatív is, ekkor a 'finish'-nek kisebbnek kell lenni mint 'start' értéknek, és persze ilyenkor lefelé számol. ld. még a NEXT parancsot.
FUNCTION xxx (arg1 [,arg2, ...]) [AS <type>} <statements> <statements> xxx = <return value> END FUNCTION	Definiál egy hívható függvényt. Ez ugyanaz, mintha hozzáadnánk egy új függvényt futás közben az MMBasic-hez. 'xxx' a függvény neve, és a változóneveknél alkalmazott elnevezési szabályokat kell alkalmazni. A függvény típusát vagy az utótag alkalmazásával (pl. xxx\$), vagy a típust megadó AS <type> használatával tehetjük meg a függvénydefiníció végén. Például: FUNCTION xxx (arg1, arg2) AS STRING 'arg1', 'arg2', stb. a függvény argumentumai, vagy paraméterei. Tömbök megadhatók az üres zárójelek használatával. Pl. arg3(). Az argumentumok típusa a típus utótag (pl. arg1\$) megadásával, vagy az AS <type> (pl. arg1 AS STRING) paranccsal végezhető. Az argumentum lehet akár egy másik definiált függvény, vagy ugyanaz a függvény, ha rekurziót használunk. (a rekurziós verem 50 egymásba ágyazott hívást enged meg). A függvény által visszaadott értéket a függvény nevéhez rendeljük. Például:

	<pre> FUNCTION SQUARE (a) SQUARE = a * a END FUNCTION </pre> <p>Minden függvénymegadásnak egyetlen END FUNCTION utasítást kell tartalmaznia. Mikor ezt elérjük, a függvény visszaadja értékét abban a kifejezésben, ahonnan meghívtuk. Az EXIT FUNCTION parancs segítségével azonnal kiléphetünk a függvényből.</p> <p>A programban a függvény nevének és argumentumainak használata pontosan olyan mintha egy szabványos MMBasic függvényt alkalmaznánk. Például:</p> <pre>PRINT SQUARE (56.8)</pre> <p>Amikor egy függvényt meghívunk a szereplő argumentumok típusának meg kell egyeznie a függvény definíciójában szereplőkkel. Ezek az argumentumok csak a függvény belsejében léteznek.</p> <p>A függvények változó számú argumentummal hívhatók meg. Bármilyen hiányzó argumentum a függvény argumentum listájában nulla szám, vagy üres karakterfüzér lesz.</p> <p>A hívólistában lévő változók (azaz nem egy kifejezés, vagy állandó) argumentum név szerint kerülnek átadásra. Ez azt jelenti, hogy az argumentum bármilyen változása a függvényben, átkerül a hívott változóba. A tömbök átadásakor a tömb nevét kell megadni üres zárójelekkel (pl. arg ()) és mindig név szerinti paraméterátadás történik.</p> <p>Nem lehet sem be, sem kiugrani egy függvényből a GOTO, GOSUB, stb. használatával. Ha mégis meg tesszük, meghatározatlan mellékhatásokra számíthatunk, beleértve annak lehetőségét, hogy tönkretesszük vele emiatt az egész napunkat.</p>
GOSUB target	Egy szubrutinhívást indít a 'target' pontra, ami lehet egy sorszám vagy egy címke. A szubrutin végét kötelezően egy RETURN zárja.
GOTO target	Elugrik a program végrehajtásához a 'target' helyre, ami lehet egy sorszám vagy egy címke.
GUI BITMAP x, y, bits [, width] [, height] [, scale] [, c] [, bc]	<p>Egy bittérkép bitjeit jeleníti meg az LCD panelen x,y ponttól kezdődően. 'height' és 'width' a bittérkép magassága és szélessége, ahogy majd az megjelenik az LCD kijelzőn, alapértelmezés szerinti mérete: 8x8 pixel.</p> <p>'scale' opcionális és alapértelmezése megegyezik a FONT parancsban beállítottakkal.</p> <p>'c' a rajzoló szín, 'bc' a háttér színe. Opcionálisak, ha nem adjuk meg, akkor az aktuális elő- és háttérszín lesz használva..</p> <p>A bittérkép lehet egész, vagy karakterfüzér változó, vagy állandó, és rajzoláskor a legfelső sorba kerülnek az első bájt bitjei (bit7 az első képpont bit6 a második, stb.), majd a második bájt, stb. Mikor a sor megtelt, a kitöltés a második sor elején folytatódik.</p> <p>Lásd az "Rajzoló parancsok" fejezetet a színek megadásához, és a grafikus koordináták értelmezéséhez.</p>
GUI CALIBRATE	Ez a parancs használható az LCD panel érintőpaneljének a megfelelő előállításához. A képernyőn kis célpontokat jelenít meg, amit pontosan a közepén kell megérinteni.
GUI RESET LCDPANEL	A konfigurált LCD-panel újraindítása. Az inicializálás automatikusan megtörténik, amikor a Micromite elindul, de bizonyos körülmények között meg kell szakítani az áramellátást az LCD panelen (pl. Az akkumulátor töltöttségének megtakarítása érdekében), és ez a parancs újraindíthatja a kijelzőt.
GUI TEST LCDPANEL vagy GUI TEST TOUCH	A kijelző vagy az érintőpanel működését teszteli az LCD panelen, GUI TEST LCDPANEL parancs végrehajtásakor egy gyorsan mozgó, ismételtlen megrajzolt színes köröket megmutató animáció jelenik meg.

	GUI TEST TOUCH parancs végrehajtásakor a képernyő törlődik, és egy érintésre vár, aminek eredményeként egy fehér pont jelenik meg az érintési helyet jelölve.
IF expr THEN statement vagy IF expr THEN stmt ELSE stmt	Kiértékeli az 'expr' kifejezést, és ha igaz, akkor végrehajtja a THEN után álló 'statement' utasítást, különben átlépi, és a következő programsorra ugrik. Az opcionális ELSE 'stmt' végrehajtása akkor következik be, ha az IF utasítás után álló kifejezés kiértékelése hamis eredményt adott. A 'THEN statement' konstrukció helyettesíthető ezzel: GOTO sorszám (vagy) címke.
IF expression THEN <statements> [ELSEIF expression THEN <statements>] [ELSE <statements>] ENDIF	Többsoros IF utasítás opcionális ELSE és ELSEIF elágazással és ENDIF-el végződik. Minden részt külön sorba kell írni. Kiértékeli az 'expression' kifejezést, és ha igaz, akkor végrehajtja a THEN után álló utasításokat, vagy ha hamis, akkor az ELSE után álló utasítások végrehajtása következik. Az ELSEIF 'expression' utasítás (ha szerepel), akkor lesz végrehajtva, ha az előző feltétel hamis, és egy új IF utasítást kezd, további ELSE és/vagy ELSEIF utasításokkal, ha szükséges. Egy ENDIF-et kell használni a többsoros IF lezárására.
INPUT ["prompt string\$";] list of variables	A konzolról változók értékét tudjuk beolvasni. Az INPUT parancs ekkor egy kérdőjellel (?) jelentkezik. A bevitelkor több változó esetén azokat vesszővel kell elválasztani. Például, ha a parancs a következő: INPUT a, b, c A billentyűzetten ezt gépeljük: 23, 87, 66 Akkor a = 23, b = 87, és c = 66 lesz. Ha "prompt string\$"-et megadjuk, akkor a kijelentkezési kérdőjel előtt ezt fogja kiírni. Ha ez a 'prompt string' vesszővel van lezárva (,) a pontosvessző helyett, a kérdőjel nem jelenik meg.
INPUT #nbr, list of variables	Hasonló az előbbihez, azzal a kivétellel, hogy a parancs a soros portról várja a változók értékeit, amit előtte INPUT-ra nyitottunk meg 'nbr'-ként. Ld. az OPEN parancsot.
IR dev, key , int vagy IR CLOSE	Dekódolja egy NEC vagy Sony gyártmányú infravörös távirányító jelét. Egy IR vevő modul használható, hogy érzékelje az IR fényt és demodulálja a jelet. Csatlakoztatni kell az IR lábra (lásd lábkiosztások). Ez a parancs automatikusan beállítja, hogy a láb bemenet legyen. Az infravörös jel dekódolása a háttérben történik, és a program tovább fut folyamatosan. "dev" és a "kulcs" változóknak numerikus kell lenniük, és értékük frissül, amikor új bemeneti jel érkezik. A "dev" a távirányító által küldött készülék kódját, míg 'key' a lenyomott távirányító gomb kódja. "int" egy felhasználó által definiált szubrutin, akkor kerül meghívásra, amikor egy új gombnyomás érkezik, vagy ha a megnyomott gomb automatikus ismétlést generál. A megszakítás szubrutinban a program megvizsgálja a 'dev' és a 'key' változók értékét és az alapján hajt végre valamilyen akciót. Az IR CLOSE parancs leállítja az IR dekódert, az I/O lábat nem konfigurált állapotba állítja. Megjegyezzük, hogy a NEC protokoll-nál a 'dev' és a 'key' bitjeinek sorrendje fordított. Például a 'key' 0. bitje a 7. bit lesz, az 1. bit a 6., stb. Ez nem befolyásolja a normál használatot, de ha egy gyártó által használt kódot kell használnunk, erre tekintettel kell lennünk. Ez megtalálható: http://www.thebackshed.com/forum/forum_posts.asp?TID=8367 linken található leírásban. Részletesen lásd a "Speciális hardver eszközök" című részt.
KEYPAD var, int, r1, r2, r3, r4, c1, c2, c3 [, c4]	Egy 4x3-as vagy 4x4 billentyűmátrixot figyel, és gombnyomást kódolja.

vagy KEYPAD CLOSE	<p>A billentyűzet figyelése a háttérben folyik, a program folyamatosan tovább fut a parancs kiadása után.</p> <p>"var" egy változó érték, ami egy gombnyomás érzékelésekor frissítésre kerül.</p> <p>'int' a felhasználó által definiált megszakítást kiszolgáló szubrutin kezdőcíme, amit egy új billentyűnyomás érzékelése indít el. Ebben a programban történik meg a 'var' változó vizsgálata és a hozzá tartozó akció végrehajtása.</p> <p>r1, r2, r3 és r4 lábszámokhoz tartozó kivezetések a billentyűzetmátrix sora-ira vannak kötve, míg a c1, c2, c3 és c4 az oszlop bekötések. c4 opcionális és csak a 4x4 mátrixnál használjuk. A parancs végrehajtásakor a lábak konfigurálása automatikusan megtörténik</p> <p>Egy gombnyomáshoz rendelt "var"-ban megjelenő szám a szokásos szám-gombokhoz rendelt értéke (pl a "6" gomb értéke 6) 10 a * gombot, és a 11. a # gombot azonosítja. A 4x4 billentyűzeteken az A,B,C,D gombok értéke 20-23.</p> <p>A KEYPAD CLOSE parancs lezárja billentyűzetkezelést és az I/O-ak visszaállnak a nem konfigurált alapállapotba.</p> <p>Részletesen lásd a "Speciális hardver eszközök" című részt.</p>
LET variable = expression	<p>Változóhoz rendeli az 'expression' értékét.</p> <p>LET automatikusan feltételezett, ha egy értékadást hajtunk végre.</p>
LCD INIT d4, d5, d6, d7, rs, en vagy LCD line , pos, text\$ vagy LCD CLEAR vagy LCD CLOSE	<p>Szöveget jelenít meg egy LCD karakteres kijelző modulon. Ez a parancs működik a legtöbb 1-soros, 2-soros vagy 4-soros LCD modullal, amelyek a KS0066, HD44780 vagy SPLC780 típusú kontrollerek valamelyikét tartalmazza (persze ez nem garantált).</p> <p>Az LCD INIT parancsot használjuk az LCD modul inicializálására. 'd4'-'d7' azok az I/O lábak sorszámai, amelyik a modul D4-D7 bemenetére kapcsolódik. D0-D3 bemeneteket földre kell kötni). 'rs' jelölésű láb a modul register select bemenetére kapcsolódik (néha a neve: CMD). 'en' az a láb, amelyik a modulengedélyező bemenetére kötjük. A modul R/W bemenetét leföldeljük. Ezen I/O lábakat a parancs automatikusan kimenetnek konfigurálja.</p> <p>Amikor a modul már inicializáltuk, akkor az adatokat kiírhatók az LCD paranccsal. A "line" a kijelző sorának a száma, értéke a kijelzőtől függően (1-4), és "pos" az első kiírandó karakter helye a sorban (az első hely 1). "text\$" egy karakterfüzér, amit meg akarunk jeleníteni az LCD-kijelzőn.</p> <p>'pos' lehet C8, C16, C20 vagy C40 akkor, ha előtte a sort törölni akarjuk, és a sorba kiírt szöveget középre akarjuk helyezni, a 8, 16, 20 vagy 40 karakter hosszúságú sorban. Például:</p> <p style="text-align: center;">LCD 1, C16, "Hello"</p> <p>LCD CLEAR az összes kijelzett adatot töröl a kijelzőről. LCD CLOSE paranccsal befejezhetjük az LCD használatát, és minden általa használt I/O láb visszaáll a nem konfigurált állapotába.</p> <p>Részletesen lásd a "Speciális hardver eszközök" című részt.</p>
LCD CMD d1 [, d2 [, etc]] or LCD DATA d1 [, d2 [, etc]]	<p>Ezek a parancsok egy vagy több bájtot küldenek az LCD kijelzőre, ami lehet parancs (LCD CMD), vagy adat (LCD DATA). Minden bájt egy szám 0 és 255 között, és vesszővel kell egymástól elválasztani. Az LCD-t előtte természetesen inicializálni kell az LCD INIT paranccsal (lásd fent).</p> <p>Ezekkel a parancsokkal lehet kezelni egy nem szabványos LCD-t, vagy fel lehet használni annak érdekében, hogy a speciális funkciókat, mint például a görgetést, kurzorokat és egyedi karaktereket fel tudjunk használni. Ehhez természetesen tanulmányozni kell a használt LCD modul adatlapját.</p>
LIBRARY SAVE vagy LIBRARY DELETE	<p>A library (könyvtár) a programmemória egy speciális része, ami programkódokat, pontosabban szubrutinokat, függvényeket, és C függvényeket tar-</p>

vagy LIBRARY LIST	<p>talmazhat. Ezek a programozó számára nem láthatók, de bármelyik Micromite-on futó program használhatja ezeket, és pontosan úgy működnek, mint az MMBasic beépített parancsai és függvényei. A részletes leírásért Ld. „<i>Speciális függvények és a Könyvtár</i>” című részt a kézikönyv elején.</p> <p>LIBRARY SAVE átemel mindent a normál program memóriából, tömöríti (eltávolítja a fölösleges adatokat, mint a megjegyzések és az üres sorok), és átmásolja a könyvtár területére (a fő programmemória így üres lesz). A könyvtárban lévő kódok nem lesznek láthatók a LIST vagy EDIT parancsok használatakor, és nem törlődnek, mikor egy új programot töltünk be, vagy kiejük a NEW parancsot..</p> <p>LIBRARY DELETE parancs törli a könyvtárat, és felszabadítja a felhasznált memóriát.</p> <p>LIBRARY LIST parancs kilistázza a könyvtár tartalmát.</p> <p>Megjegyzendő, hogy a könyvtárban lévő bármilyen kód, ami nem szubrutinban, függvényben van elhelyezve, azonnal végrehajtásra kerül, mielőtt a betöltött program futása elindul. Ez felhasználható állandók megadására, opciók beállítására, stb.</p>
LINE x1, y1, x2, y2 [, LW [, C]]	<p>Vonalat rajzol az ‘x1’ és ‘y1’ kezdőkoordinátájú pontból az ‘x2’ és ‘y2’ végpontig.</p> <p>‘LW’ a vonalvastagság, és csak függőleges vagy vízszintes vonalak esetén adható meg. Alapértelmezés szerinti értéke 1 ha nem adjuk meg, vagy a vonal ferde. ‘C’ egy egész érték, a vonal színét jelöli, és alapértelmezés szerint megegyezik az aktuális előtér színével.</p>
LINE INPUT [prompt\$,] string-variable\$	<p>Egy teljes sort olvas be a soros konzolról ‘string-variable\$’ fűzerbe. Ha ‘prompt\$’-ot megadjuk, először az kerül kinyomtatásra. Az INPUT-tól, eltérően a LINE INPUT egy teljes sort olvas, és nem áll meg a vesszővel elválasztott adatoknál.</p> <p>Kérdőjelet nem küld, kivéve, ha az a ‘prompt\$’-ban szerepel.</p>
LINE INPUT #nbr, string-variable\$	<p>Ugyanaz, mint fent, kivéve, hogy a bemenetet a soros kommunikációs portról olvassuk, amit korábban megnyitottuk INPUT-ra, mint ‘nbr’. Lásd az OPEN parancsot.</p>
LIST LIST ALL	<p>Programot kilistázza a soros konzolon.</p> <p>LIST önmagában kilistázza a programot, szünetet tartva minden teljes képernyő után.</p> <p>LIST ALL programot listáz szünet nélkül. Ez akkor hasznos, ha egy programot szándékozunk a Micromite-ba tölteni a PC-n futó terminál emulátor felhasználásával, mivel képes ezt a bemeneti karakterfolyamot fogadni.</p>
LOCAL variable [, variables] Ld. a DIM parancsot a részletekért.	<p>Változókat definiál szubrutinok, függvények belsejében történő használatra.</p> <p>Ez a parancs ugyanazt a szintaxist használja mint a DIM változókat hoz létre, és ezek a változók csak a szubrutinok és függvények belsejében lesznek használhatók. Megjegyzések:</p> <p>LOCAL csak a szubrutinok, függvények belsejében használható.</p>
LOOP [UNTIL expression]	<p>Egy programhurkot zár le. Ld.: DO.</p>
MEMORY	<p>Az éppen használt memória mennyiséget listázza ki. Például:</p> <p>Flash:</p> <p>21K (35%) Program (805 lines) 1K (1%) 2 CFunctions 1K (1%) 4 Saved Variables 37K (63%) Free</p> <p>RAM:</p> <p>9K (16%) 5 Variables 18K (32%) General</p>

	<p>26K (52%) Free</p> <p>Megjegyzések:</p> <ul style="list-style-type: none"> • Általános memóriát használják a soros I/O pufferek, stb. • Programmemóriát a NEW parancs törli. • A változók és az általános memóriahely törölhető számos paranccsal (pl. NEW, RUN, stb) valamint a speciális CLEAR és ERASE parancsokkal. • Memóriahasználat egész 1Kbájtokra van kerekítve. • Mikor programot töltünk, az először a RAM-ba kerül tárolásra (puffer), ami korlátozza a maximális program méretet. Az MMBasic tokenizálja a programot, mikor a flash memóriába írja, így a program végső mérete ettől megváltozhat.
NAME old\$ AS new\$	<p>Nevezzen át egy fájlt vagy egy könyvtárat a "régi \$" -ról "új \$" -ra. Mindkettő karakterfüzér.</p> <p>A könyvtár elérési útja mind a "old\$", mind az "new\$" -ben használható. Ha az utak különböznek, a "old\$" -ben megadott fájl átkerül a "new\$" -ben megadott elérési úttal a megadott fájl névvel.</p>
NEW	A programot törli a flash memóriából, és törli az összes változót.
NEXT [counter-variable] [, counter-variable], etc	<p>NEXT fejezi be a FOR-NEXT ciklus végét, ld. FOR.</p> <p>A 'counter-variable' adja meg pontosan, hogy pontosan melyik hurokhoz tartozik egymásba ágyazott hurkok esetén. Ha nincs 'counter-variable' megadva a NEXT a legbelső hurokhoz tartozik. Lehet megadni többszörös ilyen változót:</p> <ul style="list-style-type: none"> • NEXT x, y, z
ON ERROR ABORT or ON ERROR IGNORE or ON ERROR SKIP [nn] or ON ERROR CLEAR	<p>Ezek a parancsok vezérlik azt a tevékenységet, ha a program futása közben egy olyan hiba történik, amit az MMBasic felfedez: szintaxis hiba, hibás adatok, hiányzó hardver, stb</p> <p>ON ERROR ABORT esetén MMBasic egy hibaüzenetet ír ki, megszakítja a futó programot és a parancssorba tér vissza. Ez az MMBasic alapértelmezett viselkedése, mikor egy program futását elindítjuk.</p> <p>ON ERROR IGNORE esetén bármely hiba figyelmen kívül hagyható.</p> <p>ON ERROR SKIP figyelmen kívül hagy egy hibát az utasítást követő számos parancsban (ezt az "nn" számmal kell megadni). Az 'nn' opcionális, az alapértelmezett értéke 1, ha nincs megadva. Miután a parancsok száma befejeződött (hiba történt vagy nem), az MMBasic viselkedése visszatér az ON ERROR ABORT lehetőségre.</p> <p>Ha hiba történik, és azt figyelmen kívül hagyjuk/átugorjuk, a csak olvasható változó MM.ERRNO értéke nem nulla lesz, és az MM.ERRMSG\$ változóba kerül az a hibaüzenet, ami a hiba miatt keletkezett. Ez lenullázható és MM.ERRMSG\$-be egy üres string kerül az ON ERROR CLEAR parancs hatására. Ezek a változók akkor is törlődnek, amikor a program fut, és ha ON ERROR IGNORE vagy az ON ERROR SKIP parancsok valamelyikét használjuk.</p> <p>ON ERROR IGNORE használata nagyon megnehezíti a programhibák felderítését, ezért erősen ajánlott, hogy csak ON ERROR SKIP utasítást használjuk.</p>
ON nbr GOTO GOSUB target[,target, target,...]	<p>ON vagy elugrik (GOTO) vagy meghív egy szubrutint (GOSUB) amit a 'nbr' egészre kerekített értéke határoz meg. Ha 1, akkor az első célt, ha 2, akkor a második célt. Ez a cél (angolul: target), lehet egy sornak a száma, vagy egy címke.</p>
ON KEY target	Beállít egy megszakítást, amely meghívja "cél" felhasználó által definiált szubrutint, ha egy vagy több karakter várakozik a soros konzol bemeneti pufferében.

	<p>Ne feledjük, hogy a bemeneti pufferben várakozó minden karaktert ki kell olvasni a megszakítási szubrutinnak, különben újabb megszakítás automatikusan generálódik, amint a program visszatér a megszakításból.</p> <p>Ha szeretnénk kikapcsolni ezt a megszakítást, használjunk célként a numerikus nullát, azaz ON KEY 0.</p>
OPEN comspec\$ AS [#]fnbr	<p>Megnyit egy soros kommunikációs portot olvasásra és írásra. Két soros port áll rendelkezésre (COM1: és COM2:) és mindkettő egyszerre is nyitva lehet. A teljes leírást ld. példákkal kiegészítve az „A” függelékben.</p> <p>Az 'fnbr' segítségével egy portot írni és olvasni lehet a rá hivatkozó paranccsal vagy függvénnyel.</p>
OPTION AUTORUN OFF ON	<p>Utasítja az MMBasic-et, hogy automatikusan fusson a tárolt program, amikor a tokot bekapcsoljuk, vagy újraindul a WATCHDOG parancs miatt. Ezt kikapcsolja a NEW parancs, de más parancsok, amelyek megváltoztathatják a programmemóriát (EDIT, stb.) ezt a beállítást nem módosítja.</p>
OPTION BASE 0 1	<p>Azt állítja be, hogy mekkora index-el kezdődjön egy tömb, ez 0, vagy 1 lehet 0 vagy 1. Ezt természetesen bármely tömbdeklaráció előtt használni, és a tok bekapcsolásakor visszaáll az alapértelmezett 0 értékre.</p>
OPTION BAUDRATE nbr	<p>Beállítja a konzol adatátviteli sebességét 'nbr-re'. A beállítás azonnal megtörténik, és tápfeszültség kikapcsolása után is megmarad. A baud rate korlátozott az „A” mellékletben szereplő táblázatnak megfelelően COM1 portra vonatkozólag.</p> <p>A parancs használatával lehetséges működésképtelenséget okozó baud rate beállítása, és ilyenkor az MMBasic-et alaphelyzetbe kell állítani, amit az "MMBasic alapállapotba hozása" részben leírtunk. Ezután beáll az alaphelyzet: 38400 bit/sec.</p>
OPTION BREAK nn	<p>A megszakító billentyű kódját módosítja az 'nn' ASCII értékre. Ezt használjuk egy futó program megszakítására.</p> <p>Bekapcsoláskor ez CTRL és C gomb együttes megnyomása, de ez megváltoztatható, bármilyen klaviatúra billentyűre (például, OPTION BREAK 4 parancs után ez CTRL és D együttes megnyomása lesz.</p> <p>Ha 'nn' értékének nullát adunk, a megszakítási lehetőséget teljes mértékben letiltjuk.</p>
OPTION CASE UPPER LOWER TITLE	<p>Megváltoztatja a kisbetűs-nagybetűs listázási beállítást a LIST parancs alkalmazásánál. Az alapértelmezett a TITLE de az MMBasic régi szabványa beállítható OPTION CASE UPPER paranccsal.</p> <p>Ez az opció megmarad a tápfesz kikapcsolása után is.</p>
OPTION COLOURCODE ON vagy OPTION COLOURCODE OFF	<p>Be vagy kikapcsolja a színes karakteres kiíratást a szerkesztő program kiemenetén. A kulcsszavak lehetnek lilák, a számok pirosak stb. Az alaphelyzet a kikapcsolt állapot.</p> <p>Megjegyzések:</p> <ul style="list-style-type: none"> • A beállítás a flash memóriában van tárolva, ezért tápfesz kikapcsolás után is megmarad. • Színes kiíratás olyan terminál emulátor használatát feltételezi, amelyik képes értelmezni a színeket leíró ESC kódokat. Tera Term esetén korrektül működik, azonban Putty azt igényli, hogy az alapértelmezett háttérszín fehér legyen. • Ha színes kiíratást használunk, akkor célszerű, hogy soros konzol adatátviteli sebességét jól megnöveljük. <p>Az amerikai formátumú COLORCODE kulcsszó is használható.</p>
OPTION CONSOLE ECHO vagy OPTION CONSOLE NOECHO	<p>A soros porton működő opcióinak (működési módjainak) a beállítására szolgál.</p> <p>NOECHO kikapcsolja a konzolon leütött és kiküldött karakterek visszairását a képernyőre. ECHO ezt újra engedélyezi. Az alapértelmezett az ECHO</p>

<p>vagy OPTION CONSOLE INVERT</p> <p>vagy OPTION CONSOLE NOINVERT</p> <p>vagy OPTION CONSOLE AUTO</p>	<p>a tok bekapcsolásakor, és az opció visszaáll ECHO-ra, ha a program bármikor visszatér a parancssorba. Ez az opció hasznos, amikor a konzolt harmadik általános célú soros portnak használjuk.</p> <p>INVERT invertálja a konzol soros adó és vevő vonalának a polaritását. Ez lehetővé teszi, hogy a konzolt közvetlenül az RS232 jelekkel működtessük átalakító nélkül. Leírása: “<i>Olcsó RS-232 illesztés</i>” az „A” mellékletben.</p> <p>NOINVERT visszaállítja a konzolt normál állapotra, ez lesz utána az alapértelmezett.</p> <p>AUTO automatikusan invertálja az adat polaritást a konzolon, a jelszinttől függően, ami a bekapcsoláskor megjelent (alacsony szint jelzi, hogy a konzol invertált lesz.). Az automatikusan átkapcsol a soros TTL és az RS232 bemenet között. Ez azonban 200 msec bekapcsolási késleltetést jelent, ha az AUTO opciót használjuk.</p> <p>Ez az opció megmarad a tápfeszültség kikapcsolása után is.</p>
<p>OPTION DEFAULT FLOAT INTEGER STRING NONE</p>	<p>A külön nem típusdefiniált változó alapértelmezett típusát állítja be. Ha az OPTION DEFAULT NONE parancsot használjuk, minden változó típusát definiálni kell.</p> <ul style="list-style-type: none"> • Programfutáskor az alapértelmezés a FLOAT az MMBasic előző verzióval való kompatibilitás érdekében.
<p>OPTION DISPLAY lines [,chars]</p>	<p>Beállítja a konzolként használt kijelző terminál jellemzőit. Mind a LIST és EDIT parancsoknak ismerniük kell ezt az információt, hogy helyesen jelenítse meg a szöveget a kijelzőn.</p> <p>'lines' a kijelző sorainak száma, és 'chars' a kijelző karakterben megadott szélessége. Alapértelmezett a 24 sors x 80 karakteres kijelző, és ha ezt megváltoztatjuk, akkor ez a beállítás tápfeszültség kikapcsolás után is megmarad.</p> <p>Ha a Micromite Plus OPTION LCDPANEL CONSOLE parancsát használjuk, a sorok száma és a soronkénti karakterszám azonnal igazodik az LCD panel jellemzőihez.</p> <p>Megjegyezzük, hogy a VT100 ASCII Video Terminal első dokumentációja a kompozit video jel helytelen specifikációját adta. Ha ilyet használ egy Micromite projektben, ellenőrizze a korrekt specifikációt a következő webcímen: http://geoffg.net/terminal.html</p>
<p>OPTION EXPLICIT</p>	<p>A program elején szerepeltetve ezt az utasítást, kikényszeríti, hogy minden változót használat előtt kötelezően deklarálni kell a DIM paranccsal. Alapértelmezés szerint ez az opció tiltott.</p>
<p>OPTION LCDPANEL ILI9341, orientation, D/C pin, reset pin [,CS pin] or OPTION LCDPANEL DISABLE</p>	<p>A Micromite-ot konfigurálja, hogy együtt tudjon működni az ILI9341 vezérlőt használó LCD panellel.</p> <p>'orientation' lehet LANDSCAPE (=fekvő), PORTRAIT (álló), RLANDSCAPE or RPORTRAIT. Ezek rövidíthetők így: L, P, RL vagy RP. Az R előtag jelöli, hogy fordított, vagy „fejjel lefelé” képet mutat a kijelző.</p> <p>'D/C pin' és 'reset pin' a Micromite I/O lábai, amit használunk. Bármelyik szabad láb használható.</p> <p>'CS pin' szintén bármelyik I/O láb lehet, de ez opcionális. Ha az érintésvezérlőt nem használjuk, ez a paraméter a parancs végéről elhagyható és az LCD kijelző CS lábát fixen GND-re kell kötni. Ha az érintésvezérlőt használjuk, akkor ezt a lábat is meg kell adni, és a Micromite I/O lábára kell kötni.</p> <p>Megjegyzés: A CPU sebességének minimum 20MHz-nek kell lennie.</p>
<p>OPTION LIST</p>	<p>Kilistázza az összes opciót, amit megváltoztattunk, eltérve az alapértelmezés szerint beállítástól, és a flashbe mentettünk. Ez a parancs akkor is hasznos, amikor opciót használunk, ami I/O lábakat foglal le (pl. OPTION</p>

	LCDPANEL, vagy OPTION TOUCH) és tudni szeretnénk melyik lábakat használtuk.
OPTION PIN nbr	<p>A "nbr"-el egy PIN kódot (Personal Identification Number) állítunk be, hogy elérhessük a parancssort. "nbr" egy nem nulla szám lehet, legfeljebb nyolc számjegyig.</p> <p>Amikor egy futó program bármilyen okból megpróbál kilépni a parancssorba MMBasic kérni fogja ezt a számot, mielőtt ez megtörténne. Ez egy biztonsági funkció, mivel a parancssorhoz való hozzáférés nélkül a behatoló nem tudja sem listázni, sem módosítani, semmilyen módon befolyásolni az MMBasic működését. A funkció kikapcsolásához adjuk meg a nullát a PIN-kódként (azaz, OPTION PIN 0).</p> <p>Egy állandó zárat lehet alkalmazni a 99999999 PIN-kód használatával.</p> <p>Ha ezt alkalmazzuk, vagy elfelejtettük a PIN-kódot elvesztette az egyetlen megoldás, az MMBasic resetelése, amit az <i>"MMBasic alapállapotba hozása"</i> részben leírtunk (Ez persze törli a programmemóriát).</p>
OPTION RESET	Minden mentett opciót (még a PIN opciót is) alapértelmezés szerinti értékére állít.
OPTION TAB 2 4 8	<p>Beállítja, hogy a TAB billentyű használatakor hány betűközt lép át. Az alapértelmezett értéke 2.</p> <p>Ez az opciós beállítás megmarad még a tápfeszültség kikapcsolás után is.</p>
OPTION TOUCH T_CS pin, T_IRQ pin vagy OPTION TOUCH DISABLE	<p>Az MMBasic érintéskezelését állítja be, ami a csatlakoztatott LCD panelt konfigurálja.</p> <p>'T_CS pin' és 'T_IRQ pin' a Micromite I/O lábait azonosítja, ami a tokkiválasztáshoz, és az érintést jelző megszakítás kezeléséhez szükséges. (bármelyik szabad láb használható).</p>
PAUSE delay	<p>Megállítja a futó program végrehajtását "delay" ezredmásodpercig. Ez lehet nem egész szám is. Például, 0.2 jelentése 200 mikro szekundum. A maximális késleltetés 2147483647 msec (körülbelül 24 nap).</p> <p>Ne feledjük, hogy a megszakítások végrehajtásra kerülnek a szünet ideje alatt.</p>
PIN (pin) = value	<p>A parancssal egy digitális kimenetre állított "pin" láb értékét alacsony ("value" értéke nulla) vagy magas szintre állítjuk ("value" értéke nem nulla). Beállíthatjuk a kimenetet, hogy magas vagy alacsony legyen, mielőtt kimenetre konfigurálnánk, és ez a beállítás lesz az alapértelmezett kimeneti állapot, amikor a SETPIN parancsot végrehajtjuk.</p> <p>Lásd a PIN () függvényt a láb olvasására és a SETPIN parancsot a konfigurálására. Olvassuk el a <i>"Az I/O lábak használata"</i> című részt, ami általános leírást ad a Micromite bemeneti / kimeneti lehetőségeiről.</p>
PIXEL x, y [,c]	<p>Egy képpontot állít be az LCD panel kijelzőjén 'c' színre. "x" a vízszintes koordináta, és "y" a képpont függőleges koordinátája. A "c" egy 24 bites szám, amely meghatározza a színt. A "c" opcionális, és kihagyva az aktuális előtérzint fogja használni.</p> <p>Lásd a "Rajzoló parancsok" fejezetet a színek megadásához, és a grafikus koordináták értelmezéséhez.</p>
POKE BYTE addr%, byte vagy POKE WORD addr%, word% vagy POKE VAR var, offset, byte vagy POKE VARTBL, offset, byte	<p>Beír egy bájtot vagy egy szót a PIC32 virtuális memória területére.</p> <p>POKE BYTE egy bájtot pakol (azaz, 8 bitet) az 'addr%' címmel megadott helyre (ez a 'byte'.) 'addr%' -nak egésznek kell lenni.</p> <p>POKE WORD egy szót pakol (azaz, 32 bitet) az 'addr%' címmel megadott helyre (ez a 'byte'.) 'addr%' -nak egésznek kell lenni.</p> <p>POKE VAR egy bájtot ír a 'var' változó memóriacímére. 'offset' az egy ±eltolás a változó címéhez képest. Többöt is megadhatunk, mint var().</p>

	<p>POKE VARTBL beállít egy bájt az MMBasic változó táblájában. 'offset' egy \pmeltolás a változó tábla kezdetéhez képest. Fontos, hogy vessző szerepeljen a VARTBL kulcsszó után.</p> <p>A visszamenőleges kompatibilitás miatt, a régebbi POKE hiword, loword, val is elfogadott. Ebben az esetben a hiword'-ben szereplő 16 bit a cím felső fele, míg 'loword'-ban szerepel az alsó 16 bit.</p> <p>Ezt a parancsot csak a gyakorlott felhasználók alkalmazzák. A PIC32 mikrokontrollerekben minden kontrolregiszter, flash és RAM memória egyetlen címtartományba van beágyazva, ezért nem igényel a külvilágot megszólító INP vagy OUT parancsokat. A PIC32 adatlapja ezeket részletesen tartalmazza. Az ismerteti a címkiosztást, benne a RAM kezdőcíme 0xA0000000.</p>
PORT (start, nbr [,start, nbr]...) = value	<p>Az I/O lábak egy csoportjának az állapotát állítja be egyetlen paranccsal a megadott értékre.</p> <p>'start' egy I/O lábszám és a 'value' érték legalsó bitje (bit 0) határozza meg ennek a lábnak a kimeneti állapotát. Bit 1-et használjuk a 'start' plusz 1-es láb állapotának a beállítására, bit 2 hasonlóan a 'start'+2-re, egészen az összes 'nbr' számú bitre. A használt I/O lábakat egymás után kell számozni, és bármely érvénytelen, vagy nem kimenetre konfigurált I/O láb használata hibaüzenetet generál. A start/nbr párok ismételhetők, ha újabb bits csoportot akarunk hozzáadni.</p> <p>Például: PORT(15, 4, 23, 4) = &B11000001 8 I/O lábat állít be: a 15. láb magas lesz, míg a 16, 17, 18, 23, 24 alacsony és végül 25. és 26. lábak 1-be lesznek állítva.</p> <p>A parancs nagyon jól használható párhuzamos eszközök, például LCD kijelzők esetén. Bármelyik I/O láb 1 és a tokon lévő I/O lábak száma között használható.</p> <p>Ld. a PORT függvényt, amivel, lábcsoportok bemeneteinek párhuzamos olvasását lehet végrehajtani.</p>
PRINT expression [[,;]expression] ... etc	<p>A soros konzolra szöveget ír ki. Többszörös kifejezések használhatók, és egymástól elválaszthatók:</p> <ul style="list-style-type: none"> • Vessző (,) ami kiküld egy tab karakter • Pontosvessző (;) ami nem küld ki semmit, csak a kifejezések szétválasztására használható • Semmi vagy betűköz – ugyanaz, mint a pontosvessző. <p>A pontosvessző (;) a kifejezéslista végén elnyomja az automatikus CR/LF (kocsivissza/soremelést) print utasítás végén.</p> <p>Amikor nyomtatunk, a pozitív szám elé egy betűközt, a negatív elé egy – jelet tesz, az egész számok tizedespont nélkül nyomtatódnak, míg egyéb számok tizedesponttal és a tizedes jegyekkel. A nagy lebegőpontos számok (több mint hat számjegy) kitevős alakban kerülnek nyomtatásra.</p> <p>A TAB () függvényt oszlopos formátumú nyomtatásra lehet használni és fűzőfüggvényekkel lehet formázni a karaktersorozatokat.</p>
PRINT #nbr, expression [[,;]expression] ... etc	<p>Ugyanaz, mint fent, kivéve, hogy a kimenet előzőleg az 'nbr' számmal megnyitott soros kommunikációs portra irányítjuk. Ld.: OPEN parancs.</p>
PULSE pin, width	<p>A 'pin' lábon egy impulzust generál 'width' msec hosszúsággal, ami törtszám is lehet. Például, 0.01 egyenlő 10 μsec-el, és ez nagyon keskeny impulzus előállítását is lehetővé teszi. A minimum 5 μsec 40 MHz-es, és 40μsec 5 MHz-es órajelnél.</p> <p>A generált pulzus ellentétes állapotú, mint a használt I/O láb alaphelyzete. Például, ha kimenetet magasra van állítva, akkor a PULSE parancs egy lefutó éllel kezdődő impulzust generál. Megjegyzések:</p> <ul style="list-style-type: none"> • 'pin' lábat előzően kimenetnek kell beállítani.

	<ul style="list-style-type: none"> • A 3 msec impulzusszélességnél kisebb pulzus pontossága $\pm 1 \mu\text{sec}$. • A 3 msec impulzus szélességnél szélesebb pulzus pontossága $\pm 0.5 \text{ msec}$. <p>A 3 msec-nál szélesebb impulzusnál a generálás háttérben fut. Egyszerre max. öt különböző egyidejű impulzusgenerálás futhat a háttérben, és mindegyik ideje megváltoztatható egy új PULSE paranccsal, vagy leállítható olyan PULSE paranccsal, amiben a PULSE parancs 'width' paramétere nulla.</p>
PWM 1 , freq, 1A vagy PWM 1 , freq, 1A, 1B vagy PWM 1 , freq, 1A, 1B, 1C vagy PWM 2 , freq, 2A vagy PWM 2 , freq, 2A, 2B vagy PWM channel , STOP	<p>Pulzusszélesség modulált (pulse width modulated (PWM)) kimenetet generál analóg áramkörök számára, de hang kimenetként is használható, stb.</p> <p>Összesen egyszerre öt adott kimenet lehet PWM modulált, és ezeket a kézikönyv elején lévő tokbekötések alapján azonosíthatjuk (ezek használhatók a SERVO parancsnál is). Controller 1-nek lehet egy, kettő, vagy három PWM kimenete, míg controller 2-nek csupán egy vagy kettő. Mindkét vezérlő egymástól független, csatornáik be- és kikapcsolhatók, és különböző frekvenciájuk lehet.</p> <p>'1' vagy '2' a vezérlők sorszáma, 'freq' a kimeneti frekvencia (20 Hz – 500 kHz között). 1A, 1B és 1C a kitöltési tényezők a controller 1 kimenetén, míg 2A és 2B ugyanezek controller 2 kimenetén. A megfelelő I/O lábak automatikusan vannak kimenetként konfigurálva, míg a többi labra ez nincs hatással.</p> <p>Az összes kimenet kitöltési tényezője egymástól független, és értéküket százalékban kell megadni. Ha ez közel nulla, az impulzusok szélessége nagyon keskeny, ha 50, akkor szimmetrikus négyszöghullám jelenik meg, 100 közelében nagyon szélesek lesznek a pozitív impulzusok. 25 kHz alatt a PWM pontossága 0.1%.</p> <p>Programfutás közben a kimenetek a háttérben folyamatosan működnek, és megállíthatók a STOP paranccsal. A frekvencia és a kitöltési tényező bármikor megváltoztatható (a kimenet leállítása nélkül) egy új PWM parancs kiadásával.</p> <p>A PWM átveszi a specifikált kimenetek kezelését, és ha leállítjuk, a lábak visszaállnak magas impedanciájú, nem konfigurált állapotukra.</p>
RANDOMIZE nbr	<p>Inicializálja a véletlen szám generátort 'nbr'-el.</p> <ul style="list-style-type: none"> • Bekapcsoláskor a véletlen szám generátor nulla értékkel van inicializálva, és ezért mindig ugyanazt véletlen szám sorozatot generálja. Ha különböző véletlen szám sorozatot szeretnénk létrehozni, különböző 'nbr' értékeket kell megadni. (a TIMER függvény erre nagyon alkalmas).
RBOX x1, y1, w, h [, r] [,c] [,fill]	<p>Egy lekerekített sarkú dobozt rajzol x1, y1 pontról 'w' pixel szélesen 'h' pixel magasan a csatlakoztatott LCD panelen.</p> <p>'r' sarkoknál lévő lekerekítések sugara, alapértelmezett értéke:10.</p> <p>'c' a vonalak színe, ha nem adjuk meg, akkor az alapértelmezés szerinti előtérszín.</p> <p>'fill' a dobozt kitöltő szín. Elhagyható, vagy -1-re állítva a doboz nem lesz színnel kitöltve.</p> <p>Lásd a "Rajzoló parancsok" fejezetet a színek megadásához, és a grafikus koordináták értelmezéséhez.</p>
READ variable[, variable]...	<p>A DATA utasításban tárolt értékeket sorban kiolvastva a megadott nevű változóba tölti. A READ utasításban szereplő változótípusoknak meg kell egyeznie a DATA utasításban szereplő típusokkal kiolvasáskor. Ld. még: DATA és RESTORE.</p>
REM string	<p>REM teszi lehetővé megjegyzések elhelyezését a programban</p> <p>Az MMBasic támogatja és javasolt a Microsoft stílusú egyszeres idézőjel használata (') egy sorban a megjegyzés kezdetének a megjelölésére.</p>

RESTORE [line]	<p>A READ utasításban reseteli a sor és a pozíció számlálókat.</p> <p>Ha 'line' meg van adva, a számlálók törlődnek, az adott sor kezdetére mutatnak. A 'line' lehet egy sor száma, vagy egy címke.</p> <p>Ha a 'line' hiányzik, a számlálók törlődnek, és a program kezdetére mutatnak.</p>
RETURN	<p>RETURN befejező utasítása egy GOSUB-bal meghívott szubrutinnak, és visszatér a GOSUB-ot követő utasítás végrehajtására.</p>
RTC GETTIME vagy RTC SETTIME year, month, day, hour, minute, second vagy RTC SETREG reg, value vagy RTC GETREG reg, var	<p>RTC GETTIME kiolvassa az aktuális időt és dátumot a következő óra IC-k valamelyikéből: PCF8563, DS1307, DS3231, DS3232 és az MMBasic belső órájába írja. A dátum/idő értékeket a DATE\$ és TIME\$ függvények tartalmazzák.</p> <p>RTC SETTIME a dátum/időt az óratokba írja. Az óra 0- 23 közötti (24 órás mód).</p> <p>Az RTC SETREG és GETREG parancsok óra IC belső regisztereinek írására és olvasására szolgálnak. A 'reg' a regiszter sorszáma, 'value' a regiszterben tárolt szám 'var' egy változó, ami a regiszterből kiolvasott értéket fogja tartalmazni. Ezek a parancsok a normál működéshez nem szükségesek, de hozzáférhetünk és állíthatjuk a tok speciális képességeit (riasztás, kimeneti jelek stb.). Nagyon hasznosak a segédinformációk tárolására az RTC óra IC teleppel védett RAM memóriája.</p> <p>Ezek az RTC áramkörök I²C buszos eszközök, két felhúzó ellenállással csatlakoznak a Micromite I²C lábaira. Ha az I²C buszt már megnyitottuk, az RTC parancsok használhatók az aktuális beállításokra, egyébként előtte meg kell nyitni az I²C kapcsolatot 100 kHz sebességgel.</p> <p>Részletesen lásd a <i>"Speciális hardver eszközök"</i> című részt.</p>
RUN	<p>Futtatja a flash memóriában lévő programot.</p>
SELECT CASE value CASE testexp [[, testexp] ...] <statements> <statements> CASE ELSE <statements> <statements> END SELECT	<p>Több csoportos utasításkódból egyet hajt végre a kifejezés értékétől függően.</p> <p>A 'value' a tesztelt kifejezés. Ez lehet egy szám, vagy füzerváltozó, vagy akár összetett kifejezés is.</p> <p>A 'testexp' egy érték, amivel az 'exp' össze lesz hasonlítva. Ez lehet:</p> <ul style="list-style-type: none"> Egyszerű kifejezés (pl.34, "string" vagy PIN(4)*5) amivel egyenlő lehet Egy értéktartomány, ahol a két egyszerű kifejezést a "TO" kulcsszóval kapcsolunk össze (pl. 5 TO 9 vagy "aa" TO "cc"). Egy összehasonlítás, amit az "IS" kulcsszó vezet be (ez opcionális). Pl.: IS > 5, IS <= 10. <p>Mikor a vizsgált 'testexp' kifejezés vesszővel elválasztott számokból áll, akkor a hozzátartozó CASE utasítás végrehajtásra kerül, ha a bármelyik vizsgált kifejezés kiértékelése igaz értékű. (VAGY kapcsolatban vannak)</p> <p>Ha 'value' nem egyezik a 'testexp'-el, akkor automatikusan a CASE ELSE ág kerül végrehajtásra. Ha ez nincs megadva, akkor a program automatikusan END SELECT után álló utasítással folytatódik.</p> <p>Ha egyezés van, akkor a CASE után álló első <statements> utasítások hajtódnak végre, majd a CASE ELSE kihagyásával az END SELECT, vagy egy újabb CASE ha ez szerepel.</p> <p>Korlátlan számú CASE utasítást használhatunk, de csak egyetlen CASE ELSE ág lehet a lezáró END SELECT előtt.</p> <p>Minden SELECT CASE-nek egyetlen hozzá tartozó END SELECT lezárása lehet. Korlátlan számú SELECT...CASE utasítás ágyazható egymásba a CASE utasítás belsejében.</p> <p>Példa:</p>

	<pre> SELECT CASE nbr% CASE 4, 9, 22, 33 TO 88 statements CASE IS < 4, IS > 88, 5 TO 8 statements CASE ELSE statements END SELECT </pre>
SERVO 1 [, freq], 1A vagy SERVO 1 [, freq], 1A, 1B vagy SERVO 1 [, freq], 1A, 1B, 1C vagy SERVO 2 [, freq], 2A vagy SERVO 2 [, freq], 2A, 2B vagy SERVO channel, STOP	<p>Pozitív amplitúdójú folyamatos impulzus sorozatot generál szervók meghajtásához.</p> <p>A Micromite-nak két szervó vezérlője van, az első három a második két szervó vezérlésére képes. Mindkettő független egymástól, ki- és bekapcsolhatók, és különböző frekvenciájúak lehetnek. Ugyanazokat a lábakat használják, mint a PWM vezérlők, és ezeket a kézikönyv elején lévő tokbekötések alapján azonosíthatjuk (ezek használhatók a PWM parancsnál is). A két parancs nagyon hasonló.</p> <p>'1' vagy '2' a két vezérlő sorszáma. 'freq' a kimeneti frekvencia (20 Hz és 1000 Hz között) és ez opcionális. Ha nem adjuk meg, akkor az alapértelmezett frekvencia 50 Hz.</p> <p>1A, 1B és 1C az impulzus szélességek kontroller 1 kimenetén, míg 2A és 2B ugyanezek kontroller 2 kimenetén. A megfelelő I/O lábak automatikusan vannak kimenetként konfigurálva, míg a többi lábra ez nincs hatással.</p> <p>Az összes kimenet impulzus szélessége egymástól független, és értéküket milliszekundumban kell megadni, és ez lehet nem egész (pl. 1.345). A pontosabb pozicionálás érdekében a felbontás 0.005 msec. A minimális érték 0.001 msec, a maximum 18.9 msec. A legtöbb szervó a 0,8 msec-2.2 msec tartományban működik.</p> <p>Programfutás közben a kimenetek a háttérben folyamatosan működnek, és megállíthatók a STOP parancssal. Az impulzus szélessége bármikor megváltoztatható (a kimenet leállítása nélkül) egy új SERVO parancs kiadásával.</p> <p>A SERVO átveszi a specifikált kimenetek kezelését, és ha leállítjuk, a lábak visszaállnak magas impedanciájú, nem konfigurált állapotukra.</p> <p>Részletesen lásd a <i>“Speciális hardver eszközök”</i> című részt.</p>
SETPIN pin, cfg [, option]	<p>A 'pin' számú külső I/O lábat állítja be. Részletesen lásd a <i>“Az I/O lábak használata”</i> című részt a Micromite be- és kimeneti képességeinek a bemutatására.</p> <p>A 'pin' a konfigurálandó láb száma, 'cfg' az a mód, amire a lábat beállítjuk, 'option' egy esetleg megadható paraméter. A 'cfg' kulcsszó az alábbiak valamelyike lehet:</p> <p>OFF Nem konfigurált, inaktív</p> <p>AIN Analog bemenet (pl. a bemeneten lévő feszültség mérésére)</p> <p>DIN Digitális-bemenet</p> <p>Ha 'option' hiányzik, a bemenet magas impedanciájú lesz.</p> <p>Ha 'option' a "PULLUP" kulcsszó, akkor egy belső ellenállás kapcsolódik a tápfesz és a láb közé, vagyis a bemenet 3.3V-ra húzza fel. Ha a kulcsszó "PULLDOWN" akkor az ellenállás a föld és az I/O láb közé kerül, lehúzza a bemenetet nulla feszültségre. A fel- és a lehúzó ellenállások értéke kb. 100K.</p> <p>FIN Frekvencia-bemenet</p> <p>'option' használható a kapuzási idő megadására (az az idő, amíg számoljuk a bejövő ciklusokat). Értéke bármekkora lehet 10 msec-100000 msec között. Fontos, hogy ilyenkor a PIN() függvény visszatéréskor mindig Hz-ben adja meg a frekvenciát, a kapuzási időtől függetlenül. Ha az 'option' paramétert elhagyjuk, a kapuidő 1 másodperc lesz.</p>

	<p>PIN Periódusidő mérő bemenet</p> <p>Az 'option' használható annak a megadására, hogy hány bemeneti ciklus átlagolásával kapjuk meg az eredményt. Ez bármekkora lehet 1 és 10000 között. Fontos, hogy ilyenkor a PIN() függvény az átlagos periódusidőt adja vissza msec-ben, függetlenül az átlagolt ciklusok számától. Ha az 'option' hiányzik, egyetlen periódust használunk.</p> <p>CIN Számláló bemenet</p> <p>DOUT Digitális-kimenet</p> <p>'option' lehet "OC", ilyenkor a kimenet nyitott kollektoros lesz (pontosabban open drain). A PIN() és PORT() függvények használhatók a kimeneti láb(ak) állapotának a beolvasására is. Az MMBasic előző verziójánál 'cfg' lehetett OOUT. A visszamenőleges kompatibilitás érdekében ez is használható.</p> <p>Ld. a PIN() függvény leírását a bemenetek olvasására és a PIN()= utasítást a kimenet beállítására. Ld. a következő parancsot a megszakítás beállítására.</p>
SETPIN pin, cfg, target [, option]	<p>A 'pin' lábat konfigurálja, megszakítás generálására az 'cfg'-ben szereplő módon. A Micromite tíz lába használható megszakítás generálására.</p> <p>'cfg' egy kulcsszó, és az alábbiak valamelyike lehet::</p> <p>OFF Nem konfigurált vagy inaktív</p> <p>INTH Megszakítás alacsony-magas bemenetnél</p> <p>INTL Megszakítás magas-alacsony bemenetnél</p> <p>INTB Megszakítás mindkét váltás valamelyikénél (pl. bármely bemeneti állapotváltáskor)</p> <p>'target' a felhasználó által definiált rutin, ami akkor kerül meghívásra, ha a megszakítási esemény megtörténik. A megszakításból való visszatérés az END SUB vagy EXIT SUB parancsok valamelyikének a végrehajtásával történik.</p> <p>If 'option' a "PULLUP" kulcsszó, akkor a bemenet belső felhúzó ellenállást használunk, így a bemenet magas szintű lesz. "PULLDOWN" esetén a láb nullára lesz húzva a belső ellenállással. (100K). Ha 'option' hiányzik, a bemenet magas impedanciájú lesz.</p> <p>Ez a parancs a lábat mindig digitális bemenetnek konfigurálja, így a láb állapota a PIN() függvénnyel mindig kiolvasható.</p> <p>Részletesen lásd a <i>"Az I/O lábak használata"</i> című részt a Micromite be- és kimeneti képességeinek a bemutatására.</p>
SETTICK period, target [, nbr]	<p>Ez a parancs egy periodikus megszakítást (más néven: "tick"-et generál. Négy tick időzítő áll rendelkezésre ('nbr' = 1, 2, 3, 4). 'nbr' opcionális, ha nem adjuk meg, akkor az 1-es jelűt használjuk.</p> <p>A megszakítások közötti 'period' milliszekundum, és 'target' annak a megszakítási rutinnak a címe, ami akkor lesz aktív, ha a periódusidő lejárt.</p> <p>A periódus ideje 1 és 2147483647 msec (kb. 24 nap) között lehet.</p> <p>Ezek a megszakítások tilthatók, ha a 'period' értékét nullára állítjuk (pl. SETTICK 0, 0, 3 letiltja a 3-as tick időzítőt).</p>
SUB xxx (arg1 [,arg2, ...]) <statements> <statements> END SUB	<p>Egy meghívható szubrutint jelöl. Ez ugyanaz, mintha egy új parancsot adnánk az MMBasic-hez.</p> <p>'xxx' a szubrutin neve, és a változók elnevezési szabályai szerint kell megadni.</p> <p>'arg1', 'arg2', stb. a szubrutin paraméterei vagy argumentumai. Tömb megadásakor üres kerek zárójelpárt kell használni a név mellett. pl. arg3(). Az argumentumok típusa megadható típus utótagokkal (pl. arg1\$), vagy az AS <type> kifejezéssel (pl. arg1 AS STRING) .</p>

	<p>Minden szubrutinnak egyetlen END SUB befejezése lehet. Mikor ezt a parancsot végrehajtjuk, a program a meghívott szubrutin utáni utasítással folytatódik. Az EXIT SUB utasítást a szubrutinból való korai kilépésre használjuk.</p> <p>A szubrutinokat nevükkel, paramétereikkel együtt pontosan úgy használjuk, mint egy normális parancsot. Például: MySub A1, A2</p> <p>Amikor a szubrutint meghívunk, a meghívásban szereplő minden argumentumnak illeszkednie kell a szubrutin definíciójában szereplőkkel. Ezek az argumentumok csak a szubrutint belsejében léteznek. Szubrutinok hívhatók változó számú argumentumokkal is. Bármilyen kihagyott argumentum helyére nulla, vagy egy üres füzér kerül.</p> <p>A hívó listában szereplő argumentumok, amelyek változók (azaz nem egy kifejezés, vagy állandó) név szerint kerül átadásra a szubrutinba. Ez azt jelenti, hogy bármilyen változás a szubrutinban használt adott argumentumban át lesz másolva a hívónál szereplő változóba, ezért hozzá lehet férni a szubrutint befejezése után is. A tömbök átadásakor a tömb nevét üres zárójelekkel jelöljük (pl arg ()), és mindig név szerint történik. Az argumentum listák zárójelbe zárása a hívó helyén, illetve a definíció helyen opcionális, nem kötelező.</p>
TEMPR START pin [, precision]	<p>A 'pin' lábra kötött DS18B20 hőmérsékletérzékelő konverzióját indítja el. Fontos! Csak egy betűköz szerepelhet a TEMPR és START között.</p> <p>Normál esetben csupán a TEMPR() függvény önmagában elég a hőmérsékletmérés végrehajtásához, így a parancs használata opcionális és elhagyható.</p> <p>A parancs indítja el a hőmérsékletmérést. A program tovább fut, amíg a mérés végrehajtódik, és később a TEMPR() függvényt használjuk az eredmény kiolvasására. Ha az előtt használjuk a TEMPR() függvényt, mielőtt a konverzió befejeződött, a függvény vár a mérés végéig, és utána adja vissza az eredményt.</p> <p>'precision' a mérés felbontása, opcionális. Értéke 0-3 között lehet:</p> <p>0 = 0.5°C felbontás, 100 msec konverziós idő.</p> <p>1 = 0.25°C felbontás, 200 msec konverziós idő (ez az alapértelmezett).</p> <p>2 = 0.125°C felbontás, 400 msec konverziós idő.</p> <p>3 = 0.0625°C felbontás, 800 msec konverziós idő.</p>
TEXT x, y, string\$ [, justification] [, font] [, scale] [, c] [, bc]	<p>Az LCD panelen egy szöveget jelenít meg x,y kezdőponttal.</p> <p>'string\$' a megjelenítendő karaktersorozat. Megjegyezzük hogy a numerikus adatok karaktersorozattá konvertáló és formázható a Str\$() függvény felhasználásával.</p> <p>'justification' egy vagy két betűs karakterfüzér, ahol az első betű a vízszintes igazítást jelöli x ponttól, és lehet L, C vagy R (LEFT - bal, CENTER - középp, RIGHT - jobb) A második betű függőleges elhelyezkedést jelöli y koordinátához képest, és lehet: T, M vagy B (TOP - fent, MIDDLE - középp, BOTTOM - lent). Az alapértelmezés szerinti igazítás: bal/fent.</p> <p>A Micromite Plus-ban egy harmadik karakter használható az igazításban a szöveg forgatásának jelzésére. Ez lehet "N" a normál tájoláshoz, "V" a függőleges szöveghez, minden egyes karakternél az előzőhöz képest fentről lefelé, "I" a szöveg fordítva (azaz fejjel lefelé), "U" a szöveg lesz Az óramutató járásával ellentétes irányban 90 ° -kal elforgatva, a "D" pedig a szöveg 90°-os elforgatásával történik</p> <p>'font' és 'scale' opcionálisak (megadhatók), alapértelmezésüket a FONT parancs állítja be.</p> <p>'c' a rajzoló szín, 'bc' a háttérszín. Ezek is opcionálisak, alapértelmezésük az aktuális elő és háttérszín.</p>

	Lásd az "Rajzoló parancsok" fejezetet a színek megadásához, és a grafikus koordináták értelmezéséhez.
TIMES = "HH:MM:SS" vagy TIMES = "HH:MM" vagy TIMES = "HH"	Beállítja a belső óra idejét. MM és SS megadása opcionális, és ha nem adjuk meg az alapértelmezés szerinti értéke nulla lesz. Például TIMES = "14:30" az órát 14:30-ra állítja, nulla másodperccel. Bekapcsoláskor az idő "00:00:00"-ra lesz állítva.
TIMER = msec	Az időzítőt megadott 'msec'-re állítja. Normálisan ezt az időzítő nullázására használjuk, de bármilyen pozitív számú msec értékre állíthatjuk. Részletekért lásd a TIMER függvényt.
TRACE ON vagy TRACE OFF vagy TRACE LIST nn	TRACE ON/OFF be- illetve kikapcsolja a nyomkövető képességet. Ez a képesség kinyomtatja minden végrehajtott sor sorszámát (a program kezdetétől sorszámozva) szögletes zárójelbe zárva. Programok hibakeresésénél hasznos. Csak a Micromite Plus-nál: TRACE LIST nn kilistázza az utolsó nn sorban lévő utasítás sorszámát. Megjegyzendő, hogy az MMBasic mindig megjegyzi a végrehajtott sorokat, így ez a tulajdonság mindig kihasználható, azaz ezt nem kell külön bekapcsolni.
VAR SAVE var [, var]... vagy VAR RESTORE vagy VAR CLEAR	VAR SAVE egy vagy több változót elment a nem felejtő flash memóriába, ahol később visszaállíthatók (általában pl. áramkimaradás után). 'var' lehet tetszőleges számú numerikus vagy füzérváltozó és/vagy tömb. Tömbök megadásánál üres zárójelet kell használni. (Pl.: var()). VAR RESTORE visszatölti a korábban mentett változókat, visszahelyezi értéküket a változó listába. A VAR SAVE parancs ismételve is használható. Azok a változók, amelyeket már előzetesen mentettük, frissítve lesznek az új, aktuális értékükkel, és minden új (előzőleg nem mentett) változó hozzáadódik az aktuális mentett változók listájához. VAR CLEAR utasítás minden mentett változót töröl. Akkor is megtörténik a törlés, ha kiadunk egy NEW parancsot, vagy ha egy új programot töltünk be. Ezt a parancsot általában kalibrációs adatok, beállítások és egyéb adatok mentésére, használják, melyek nem változnak gyakran, de meg kell tárolni az értéküket egy áramkimaradás esetén. Normális esetben a VAR RESTORE parancsot a program elején szerepeltetjük, hogy a korábban mentett változók azonnal elérhetők legyenek a program indulásakor. Megjegyzések: <ul style="list-style-type: none"> • A rendelkezésre álló tárhely ennél a parancsnál 2 KiB, illetve 4 KiB Micromite Plus esetén. • Ha VAR RESTORE parancsot használjuk és előtte nem volt mentés, semmi nem történik, és hibát sem okoz. • Ha használjuk a VAR RESTORE, és ha egy változó már ugyanolyan névvel létezik, akkor értéke felülíródik. • A mentett tömböket előzetesen deklarálni kell (DIM-et használva), mielőtt vissza kívánjuk állítani értéküket. Figyeljünk arra, hogy a karakterfüzér tömbök gyorsan elfoglalják a használható memóriát. A LENGTH hosszjelző használható a karaktertömbök megadásánál, hogy csökkentjük a tömb méretét. (ld. DIM parancs)
WATCHDOG timeout vagy WATCHDOG OFF	Elindítja a watchdog időzítőt, amely automatikusan újraindítja a processzort, ha időtűllépés következett be. Ezt fel lehet használni, arra hogy a program automatikusan újrainduljon, ha a program valamilyen nem fut tovább (például egy végtelen ciklusba kerül, vagy a programozói vagy egyéb

	<p>hiba, ami leállítja a programot. Ez fontos lehet egy nem kívánt működési szituáció kezelésére.</p> <p>A 'timeout' az az idő, ezredmásodpercben (msec), mielőtt a számítógép újraindítását kikényszeríti. Ezt a parancsot kell elhelyezni a futó program "stratégiai helyein" hogy mindig törölje a watchdog számlálót, ami megakadályozza, hogy túlsordulásával újraindítsa a programot.</p> <p>Ha az időzítő eléri a nullát (talán azért, mert az alapprogram leállt) a Micromite automatikusan újraindul, és az automatikus változó MM.WATCHDOG 1-be állítódik be, jelezve, hogy hiba történt. Bekapcsolás utáni indításkor MM.WATCHDOG értéke 0 lesz.</p> <p>Bármikor használható a WATCHDOG OFF parancs, amivel le lehet tiltani a watchdog időzítőt (ez az alapértelmezett tápfeszültség bekapcsoláskor). Az időzítő akkor is kikapcsol, ha a break karaktert (általában a CTRL-C) adunk ki a konzolon a már futó programot leállítására.</p>
<p>XMODEM SEND vagy XMODEM RECEIVE</p> <p>Micromite Plus only: XMODEM SEND, file\$ vagy XMODEM RECEIVE, file\$</p>	<p>Küldi vagy veszi a BASIC programot egy számítógépre/ről, az XModem protokoll felhasználásával. Az átvitel a soros konzol kapcsolaton keresztül történik.</p> <p>XMODEM SEND küldi a Micromite programmemóriájának a tartalmát a távoli számítógépre.</p> <p>XMODEM RECEIVE fogad egy távoli számítógépről érkező programot, és eltárolja a Micromite programmemóriájában, az esetleges bennlévő programot felülírva. Fontos, hogy először a program a RAM memóriába kerül (pufferelődik), ami korlátozza a program maximális méretét.</p> <p>SEND és RECEIVE rövidíthető az S és R betűkkel.</p> <p>A Micromite Plus használatakor megadhatja a file\$-t is, amely átmásolja az adatokat az SD-kártyán lévő fájlról /fájlra. Ha a fájl már létezik, felülíródik a fájl fogadásakor.</p> <p>Az XModem protokoll igényel a távoli gépen futó együttműködő programot, és annak soros portját használja. A parancs a Windows alatt futó Tera Term programmal együttműködik, javasolt a használata. Ezt futtatva kiválasztjuk az XMODEM parancsot:</p> <p>File -> Transfer -> XMODEM -> Receive/Send</p> <p>a Tera Term menüből, hogy elindítsuk az átvitelt.</p> <ul style="list-style-type: none"> • A kapcsolódás ideje legfeljebb 15 másodperc, és ha az XMODEM parancs nem tud kommunikációt létesíteni, akkor visszatér az MMBasic parancssorába egy perc múlva a programmemória módosítása nélkül. A Tera Term letöltési helye: http://ttssh2.sourceforge.jp/

Függvények

Az I²C, egyvezetékes és SPI kommunikációval kapcsolatos funkciók nem szerepelnek ebben a fejezetben, a részletes leírásuk megtalálható a dokumentum végén lévő „A”, „B”, „C” és „D” függelékekben.

Szögletes zárójelek jelzik, hogy a paraméter, vagy a karakterek megadása nem kötelező, opcionális.

ACOS (number)	Visszadja a radiánban megadott 'number' érték arkusz koszinuszát
ABS (number)	Visszaadja a 'number' szám abszolút értékét.
ASC (string\$)	Visszaadja a "string \$" füzér első betűjének ASCII kódját.
ASIN (number)	Visszadja a radiánban megadott 'number' érték arkusz szinuszt
ATN (number)	Visszaadja radiánban adott 'number' szám arctg értékét.
BIN\$ (number [, chars])	Egy füzérrel tér vissza, amiben a szám ('number') bináris alakja van. 'chars' opcionális és meghatározza a karakterek számát a füzérben, és a nem használt első helyeken nullák lesznek.
CHR\$ (number)	A 'number' számhoz tartozó ASCII kód karakteres megfelelőjét adja vissza.
CINT (number)	Egészre kerekíti a számot a tizedes rész értékétől függően. Pl.: 45.47 kerekítve 45 45.57 kerekítve 46 -34. kerekítve -34 -34.55 kerekítve -35 Lásd még: INT() és FIX().
COS (number)	Visszaadja radiánban adott 'number' szám cos értékét.
DATES	Az MMBASIC belső órájában lévő dátumot adja vissza füzérként "DD-MM-YYYY" formában. Pl.: "28-07-2012". Ez a belső óra/naptár tárolja és lépteti az időt és dátumot, és figyelembe veszi a szökőévet is. A dátum beállítása a DATE\$ = paranccsal lehetséges.
DEG (radians)	'radians' értéket fokokká konvertál.
EOF ([#]nbr)	Soros kommunikációs portnál ez a függvény visszatér igaz értékkel, ha a vevő pufferben nincs várakozó karakter. A # használata opcionális. Ld. még, az OPEN, INPUT és LINE INPUT parancsokat és az INPUT\$ függvényt.
EXP (number)	A 'number' exponenciális értékével tér vissza.
FIELD\$ (str\$, field, delim\$)	Egy szövegrész (egy mező) kiemelése a "str\$" karakterfüzérből. Mindegyik részt elválasztja a "delim \$" karakterláncban szereplő karakterek valamelyike. A visszaadott mező sorszámát "field" határozza meg (az első mező az 1. mező). Bármely vezető és záró betűköz a visszaadott karakterláncban levágásra kerül. Ne feledjük, hogy a 'delim\$' számos karaktert tartalmazhat, és a mezőket ezek a karakterek bármelyike elválasztja. Ez a funkció hasznos a vesszővel elválasztott értékek (CSV) szétválasztásához amit a GPS modulok és más berendezések által előállított adatfolyamok tartalmaznak. Például: <pre>PRINT FIELD\$ ("aaa,bbb,ccc", 2, ",", " ")</pre> Utasítás kinyomtatja a "bbb" füzért.
FIX (number)	Egy számot egész számra csonkít, elhagyva a tizedespontot és az összes ez után álló számjegyet. Például a 9.89 visszatérési értéke 9 és -2.11 pedig -2-vel tér vissza.

	<p>A különbség a 'FIX' és az 'INT' között, hogy a 'FIX' egy igazi egész értéket ad vissza, előzetes kerekítés nélkül. Azaz, nem tér vissza a következő alacsonyabb negatív számmal, míg az 'INT' () igen.</p> <p>Ez a viselkedés a Microsoft kompatibilitás miatt van.</p> <p>Lásd még CINT ().</p>
HEX\$(number [, chars])	<p>Egy füzérrel tér vissza, amiben a szám ('number') hexadecimális alakja van. 'chars' opcionális és meghatározza a karakterek számát a füzérben, és a nem használt első helyeken nullák lesznek.</p>
INKEY\$	<p>Ellenőrzi a konzol bemeneti puffert és, ha egy vagy több karakter vár a sorban, eltávolítja el az első karakter, és azt küldi vissza egyetlen karakterként. Ha a bemeneti puffer üres, egy üres füzért (azaz "") küld vissza.</p>
INPUT\$(nbr, [#]fnbr)	<p>Vissza fog térni egy füzérrel, ami "nbr" karakterből áll, és amit az "fnbr". számmal megnyitott soros kommunikációs portról olvasunk. Ez a függvény annyi karakterrel fog visszatérni, amennyi várakozik a vételi pufferben maximum "nbr" számúval. Ha nincsenek karakterek a pufferben, akkor azonnal visszatér egy üres füzérrel.</p> <p>#0 használható a konzol bemenő pufferére való hivatkozásra.</p> <p>A # karakter használata opcionális. Lásd még az OPEN parancsot.</p>
INSTR([start-position,] string-searched\$, string-pattern\$)	<p>Azzal a pozícióval tér vissza, amelynél a 'string-pattern\$' füzér előfordul a 'string-searched\$' füzérben, kezdve a 'start-position'-tól.</p> <p>A visszatérő pozíció és a 'start-position' esetén 1 az első karakter sorszáma 2 a másodiké, stb. A függvény nullával tér vissza, ha a 'string-pattern\$' nem található.</p>
INT(number)	<p>Csonkít egy számot a következő egész számra, amely kisebb vagy egyenlő, mint az eredeti szám. Például 9.89 visszatér a 9-el. és -2.11 visszatér -3-al.</p> <p>Ez a viselkedés a Microsoft kompatibilitás miatt van, a FIX () függvény valószínűleg meg az igazi egész funkciót.</p> <p>Lásd még CINT ().</p>
LEFT\$(string\$, nbr)	<p>A 'string\$' egy részfüzérével tér vissza, ami 'nbr' karaktert tartalmaz a füzér bal oldaláról, vagyis az elejétől.</p>
LEN(string\$)	<p>Visszaadja a 'string\$' füzérben lévő karakterek számát.</p>
LOC([#]fnbr)	<p>Soros kommunikációs portnál, ha 'fnbr' számmal nyitottuk meg, visszatér a vett bajtok számával, és várja, hogy a vevő puffert kiolvassuk.</p> <p>#0 használható a konzol bemenő pufferére való hivatkozásra.</p> <p>A # paraméter opcionális.</p>
LOF([#]fnbr)	<p>Soros kommunikációs portnál, ha 'fnbr' számmal nyitottuk meg, visszatér az adó pufferben még fennmaradó üres karakterhelyek számával. Megjegyzendő, ha a puffer tele van, az MMBasic várakozik, ha új karaktert akarunk beírni, és vár arra, hogy hely szabaduljon fel a pufferben.</p> <p>A # paraméter opcionális.</p>
LOG(number)	<p>Visszaadja 'number' szám természetes alapú logaritmusát.</p>
LCASE\$(string\$)	<p>Visszaadja a 'string\$' füzér minden karakterét kisbetűsre alakítva.</p>
MAX\$(arg1 [, arg2 [, ...]]) vagy MIN\$(arg1 [, arg2 [, ...]])	<p>Az argumentumlistából megadja a maximális vagy minimális értékű számot. Megjegyezzük, hogy az összehasonlítás lebegőpontos összehasonlítás (az egész argumentumokat lebegőpontosá alakítjuk át), és a visszatérő érték is lebegőpontos.</p>
MID\$(string\$, start) vagy MID\$(string\$, start, nbr)	<p>A 'string\$' egy részfüzérével tér vissza, ami az eredetiben 'start'-nál kezdődik és 'nbr' karakterig folytatódik. A füzér első karaktere az 1-es számú.</p> <p>Ha 'nbr'-t elhagyjuk, a visszatérési füzér a 'string\$' végéig fog tartani.</p>

OCT\$(number [, chars])	Egy füzérrel tér vissza, amiben a szám ('number') oktális alakja van. 'chars' opcionális és meghatározza a karakterek számát a füzérben, és a nem használt első helyeken nullák lesznek.
PEEK(BYTE addr%) vagy PEEK(WORD addr%) vagy PEEK(VARADDR var) vagy PEEK(CFUNADDR cfun) vagy PEEK(VAR var, ±offset) vagy PEEK(VARTBL, ±offset) vagy PEEK(PROGMEM, ±offset)	<p>A PIC32 virtuális memóriájának egy bájtjával vagy szavával tér vissza. BYTE bájt (8 bit) tér vissza, ami az 'addr%' címen helyezkedik el. WORD szóval (32 bit) tér vissza, ami az 'addr%' címen helyezkedik el. VARADDR visszatér a 'var' változó 32 bites címével. Tömb megadásakor a var() jelölést kell használni.</p> <p>CFUNADDR visszatér a memóriában lévő 'cfun' nevű CFunction program 32 bites címével. Ez a cím átadható egy másik CFunction programnak, amely aztán meghívható, hogy néhány folyamatot végrehajtsa.</p> <p>VAR a memóriában elhelyezkedő 'var' változó bájtos tartalmát adja vissza. Tömb megadásakor a var() jelölést kell használni.</p> <p>VARTBL visszatér egy bájt az MMBasic változó táblájából. 'offset' egy ±eltolás a változó tábla kezdetéhez képest. Fontos, hogy a vessző szerepeljen a VARTBL kulcsszó után.</p> <p>PROGMEM, visszatér egy bájt, ami a programmemóriában helyezkedik el. Fontos, hogy a vessző szerepeljen a PROGMEM kulcsszó után.</p> <p>Az 'addr%'-nek egésznek kell lenni.</p> <p>Visszamenőleges kompatibilitás miatt, a régebbi PEEK hiword, loword is elfogadott. Ebben az esetben a hiword'-ban szereplő 16 bit a cím felső fele, míg loword'-ban szerepel az alsó 16 bit.</p> <p>Ezt a parancsot csak gyakorlott felhasználók alkalmazzák! A PIC32-ben minden regiszter, memória egyetlen címtartományban van elhelyezve, ezért nem igényel a külvilágot megszólító INP vagy OUT parancsokat. A PIC32 adatlapja ezeket a címtartományokat részletesen tartalmazza. A RAM a 0xA0000000 címen, a Program Flash 0x9D000000 címen és a Bootflash 0x9FC00000 címen kezdődik.</p>
PI	Pi értékét adja vissza.
PIN(pin)	<p>Egy külső I/O láb "pin" értékével tér vissza. Ha nulla, akkor a láb jelszintje digitális alacsony szintű, 1 jelenti a digitális magas szintet, analóg bemeneteken pedig a lábon mért feszültség értékével fog visszatérni, ami egy lebegőpontos szám.</p> <p>Frekvencia bemenet esetén a Hz-ben megadott frekvencia mért értékével tér vissza. Periódusidő bemenet esetén az időtartamot adja ezredmásodpercben, míg számláló bemenetnél a reszet óta események számával tér vissza. (a számlálás pozitív felfutó élnél történik). A számláló bemenet lehet nullázni, ha ismét kiadjuk a PIN(pin) parancsot. (akkor is, ha már így volt beállítva).</p> <p>Ez a funkció visszaadja a kimenetként beállított láb állapotát is.</p> <p>Ld. még a SETPIN és PIN () = parancsokat. Olvassa el a "<i>Az I/O lábak használata</i>" fejezetet, ami egy általános leírást ad a Micromite bemeneti / kimeneti lehetőségeiről.</p>
PORT(start, nbr [,start, nbr]...)	<p>Visszaadja az I/O lábak egy csoportjának az értékét egy művelettel.</p> <p>'start' egy I/O lábszám és a 'value' érték legalsó bitje (bit 0) határozza meg ennek a lábnak a bemeneti állapotát. Bit 1-et használjuk a 'start' plusz 1-es láb állapotának a beállítására, bit 2 hasonlóan a 'start'+2-re, egészen az összes 'nbr' számú bitre. A használt I/O lábakat egymás után kell számozni, és bármely érvénytelen, vagy nem bemenetre konfigurált I/O láb használata hibaiüzenetet generál. A start/nbr párok ismételhetők, ha újabb bitesortot akarunk hozzáadni.</p>

	<p>Ez a függvény visszaadja a kimenetként beállított láb állapotát is. Kényelmesen használható párhuzamos eszközökkel történő kommunikációra, például memóriachipecknél. Bármennyi I/O láb (és így bit) használható 1-től a chipen található I/O lábszámig.</p> <p>Ld. a PORT parancsot, amivel, lábcsoportok kimeneteinek párhuzamos írását lehet végrehajtani.</p>
POS	Visszaadja a kurzor aktuális pozícióját a kijelző sorában.
PULSIN (pin, polarity) vagy PULSIN (pin, polarity, t1) vagy PULSIN (pin, polarity, t1, t2)	<p>Egy bemeneti impulzus szélességét méri 1 µsec - 1 sec között 0.1µsec felbontásban.</p> <p>A 'pin' a mérésre használt I/O láb, amit előzőleg digitális bemenetre konfiguráltunk. A 'polarity' a mérni kívánt impulzus típusa, ha nulla, a függvény viszszaátér a következő negatív impulzus szélességével, ha nem nulla, megméri a következő pozitív impulzust.</p> <p>A "t1" az időtúllépés, amíg maximum várunk az impulzus érkezésére, "t2" az időtúllépés, míg mérjük az impulzust. Mindkettőt mikroszekundumban (µsec) kell megadni, és opcionális. Ha 't2' hiányzik akkor T1" értékét használjuk mindkét időtúllépés értékének. Ha mindkét "t1" és "t2" hiányzik, akkor mindkét időkorlát 100000 (azaz 100 msec) lesz.</p> <p>Ez a függvény az impulzus szélességét mikroszekundumban (µsec) adja vissza, vagy -1-et, ha időtúllépés történt. Ha a processzor sebessége 40MHz a mérés pontossága ± 0,5% és ± 0.5µsec. Más sebességeknél a mérés kevésbé pontos. Megjegyzendő, hogy ez a függvény a mérés idejére a futó programot leállítja, és a megszakításokat is figyelmen kívül fogja hagyni ebben az időszakban.</p>
RAD (degrees)	'degrees' fokokat radiánra alakít.
RGB (red, green, blue) or RGB (shortcut)	<p>Egy RGB 24 bites „true colour” értéket hoz létre.</p> <p>'red', 'blue' és 'green' jelöli mindhárom szín intenzitását. A nulla értéknek a fekete (nulla intenzitás), 255 a maximális színerősség.</p> <p>'shortcut' lehetővé teszi, hogy használjuk a színek angol neveit. Ezek: white, black, blue, green, cyan, red, magenta, yellow, brown and gray. Például: RGB(red) vagy RGB(cyan).</p>
RIGHTS (string\$, number-of-chars)	A 'string\$' egy részfüzérével tér vissza, ami 'nbr' karaktert tartalmaz a fűzér job oldaláról, vagyis a végétől.
RND (number)	Visszaad egy álvéletlen számot a 0-0.999999 tartományból. A "number" értéket figyelmen kívül hagyja, ha megadjuk. A RANDOMIZE parancs újrainicializálja a véletlenszám-generátort.
SGN (number)	Visszaadja a 'number' szám előjelét, +1 pozitív, 0 nulla, és -1 negatív szám esetén.
SIN (number)	Visszaadja radiánban adott 'number' szám sin értékét.
SPACES (number)	'number' számú betűközökből álló fűzért ad vissza.
SQR (number)	Visszaadja a 'number' szám négyzetgyökét.
STR\$ (number) vagy STR\$ (number, m) vagy STR\$ (number, m, n) vagy STR\$ (number, m, n, c\$)	<p>Egy fűzérrel tér vissza, amiben a szám ('number') decimális alakja van.</p> <p>Az 'm' opcionális és meghatározza a tizedespont előtti karakterek számát a fűzérben. A nem használt első helyeken betűközök lesznek. Hogyha az 'm' nullaértékű vagy a számban a tizedespont előtti számjegyek számánál kisebb, nem tesz betűköz karaktert a szám elé.</p> <p>Ha az 'm' negatív, egy pluszjelet rak a pozitív szám elé egy mínuszjelet a negatív szám elé és betűköz karakterekkel tölti föl a szám előtti részt.</p> <p>Az 'n' a dígitok száma a tizedespont után. A maximális értéke hét, és ha az 'n' értéke nulla, a fűzérben nem lesz tizedes érték. Ha az 'n' paramétert nem adjuk meg, a tizedespont utáni számjegyek száma a szám értékétől függően változni fog.</p>

	<p>A 'c\$' egy füzér, és ha megadjuk a füzér első karakterét, ez lesz a számok elején lévő vezető karakterek helyén a betűköz helyett (ld. az 'm' argumentumot).</p> <p>Példák:</p> <table> <tr> <td>STR\$(123.456)</td><td>visszatér "123.456"</td></tr> <tr> <td>STR\$(123.456, 6)</td><td>visszatér " 123.456"</td></tr> <tr> <td>STR\$(123.456, -6)</td><td>visszatér " +123.456"</td></tr> <tr> <td>STR\$(-123.456, 6)</td><td>visszatér " -123.456"</td></tr> <tr> <td>STR\$(-123.456, 6, 5)</td><td>visszatér " -123.45600"</td></tr> <tr> <td>STR\$(53, 6)</td><td>visszatér " 53"</td></tr> <tr> <td>STR\$(53, 6, 2)</td><td>visszatér " 53.00"</td></tr> <tr> <td>STR\$(53, 6, 2, "***")</td><td>visszatér "*****53.00"</td></tr> </table>	STR\$(123.456)	visszatér "123.456"	STR\$(123.456, 6)	visszatér " 123.456"	STR\$(123.456, -6)	visszatér " +123.456"	STR\$(-123.456, 6)	visszatér " -123.456"	STR\$(-123.456, 6, 5)	visszatér " -123.45600"	STR\$(53, 6)	visszatér " 53"	STR\$(53, 6, 2)	visszatér " 53.00"	STR\$(53, 6, 2, "***")	visszatér "*****53.00"
STR\$(123.456)	visszatér "123.456"																
STR\$(123.456, 6)	visszatér " 123.456"																
STR\$(123.456, -6)	visszatér " +123.456"																
STR\$(-123.456, 6)	visszatér " -123.456"																
STR\$(-123.456, 6, 5)	visszatér " -123.45600"																
STR\$(53, 6)	visszatér " 53"																
STR\$(53, 6, 2)	visszatér " 53.00"																
STR\$(53, 6, 2, "***")	visszatér "*****53.00"																
STRING\$(nbr, ascii) vagy STRING\$(nbr, string\$)	Visszaadja egy füzért "nbr" bájt hosszúsággal, amit feltölt vagy a string\$ paraméter első karakterével a vagy az "ascii" számérték ASCII karakteres megfelelőjével. Ez utóbbi esetben a tartomány: 32-126.																
TAB(number)	Betűközöket küld ki addig, míg a 'number' számmal jelzett oszlopot el nem érjük.																
TAN(number)	Visszaadja radiánban adott 'number' szám tangens értékét.																
TEMPR(pin)	<p>Visszatér a "pin" lábra csatlakoztatott DS18B20 hőmérsékletérzékelő által mért hőmérséklet eredményével. A lábat nem kell konfigurálni.</p> <p>A CPU minimális órajel frekvenciája: 20 MHz.</p> <p>A visszaadott érték Celsius fokban van, alapértelmezett felbontása 0.25°C.</p> <p>Ha hiba van a mérés során, a visszaadott érték 1000 lesz.</p> <p>A szükséges idő a teljes mérésre 200 msec, és ez idő alatt a CPU egyéb megszakításait figyelmen kívül hagyja. Alternatívaként a TEMPR START paranccsal lehet elindítani a mérést, és a programunk közben tovább futhat. Ha ezután meghívjuk a függvényt, a mérés értéke azonnal kiolvasható, miután a konverziós idő letelt. Amennyiben ez még nem történt meg, a függvény ki várja a fennmaradó konverziós időt, mielőtt visszatérne az értékkel.</p> <p>A DS18B20 táplálható külön is egy 3-5 V-os tápegységgel, vagy működhet a Micromite parazita áramával.</p> <p>Lásd a "<i>Speciális hardver eszközök</i>" fejezetet a további részletekért.</p>																
TIMES	<p>Füzerként visszaadja az MMBasic's belső órájának aktuális értékét "HH:MM:SS" 24 órás alakban. Pl.: "14:30:00".</p> <p>Az aktuális idő megadásához használjuk a TIME\$ = parancsot.</p>																
TIMER	<p>Visszaadja a reszet óta eltelt időt ezredmásodpercben (pl 1/1000 másodperc).</p> <p>Az időzítő nullázódik a tápfeszültség bekapcsolásakor, vagy a CPU újraindításakor, és nullázható a TIMER parancsként való alkalmazásával. Ha nem töröljük, akkor folytatja a számolást a végtelenségig (ez egy 64 bites szám, és csak 200 millió év múlva fog túlszordulni).</p>																
TOUCH(X) vagy TOUCH(Y)	Az LCD panel képernyőjén megérintett pont X vagy Y koordinátájával tér vissza. Ha nem érintettük meg a képernyőt, akkor a visszatérési érték: -1.																
UCASE\$(string\$)	Visszaadja a 'string\$' füzér minden karakterét nagybetűsre alakítva.																
VAL(string\$)	<p>A 'string\$' füzér numerikus értékével tér vissza. Ha a 'string\$' egy érvénytelen szám, a függvény nullát ad vissza.</p> <p>A függvény felismeri a &H előtagot, jelölve, hogy a szám hexadecimális érték, az &O oktális és &B bináris jelölést is ismeri.</p>																

Elavult parancsok és függvények

Ezek a parancsok és függvények nagyrészt azért vannak, hogy segítsék átalakítani a Microsoft BASIC-ben írt programokat. Az új programoknál a megfelelő MMBasic parancsokat kell használni.

Elképzeltető, hogy a jövőben ezek a parancsok és függvények megszűnnek, hogy az általuk foglalt memóriaterületet más célokra felhasználhassuk.

IF condition THEN linenbr	Microsoft kompatibilitás miatt GOTO utasítást tételezünk fel, ha a THEN utasítást egy program sorszám követi. Címke ebben az esetben nem megengedett. Új programoknál ezt használjuk: IF feltétel THEN GOTO sorszám címke
IRETURN	Visszatér a megszakításból, ha a megszakítás cílcíme sorszám vagy címke volt. Új programoknál a felhasználó által definiált szubrutint kell használni a megszakítás kiszolgálására. Ebben az esetben END SUB vagy EXIT fogja biztosítani a visszatérést a megszakításból.
SPC (number)	Ez a függvény 'number' számú betűközt tartalmazó füzérrel tér vissza. Ez hasonló a SPACE\$() függvényhez és csak a Microsoft kompatibilitást szolgálja.
TROFF	A TRON-al bekapcsolt nyomkövetést kapcsolja ki. ld. TRON
TRON	Bekapcsolja a nyomkövető képességet. Ez a képesség kinyomtatja minden végrehajtott sor sorszámát (a program kezdetétől sorszámozva) szögletes zárójelbe zárva. Programok hibakeresésénél hasznos. Új programoknál a TRACE parancsot kell használni.
WHILE expression WEND	WHILE indít egy WHILE-WEND hurkot. A hurok egy WEND-el fejeződik be, és a végrehajtás ciklikusan ismétlődik a hurokban, egészen addig, amíg az 'expression' igaz. Ezt a konstrukciót is a Microsoft BASIC nyelv kompatibilitás miatt építettük be. Új programokban használjuk a DO WHILE ... LOOP szerkezetet.

Soros kommunikáció – A melléklet

A Micromite-oknál két soros port áll rendelkezésre aszinkron soros kommunikációra, ezek jelölése COM1:, és COM2:. Megnyitásuk után egy hozzárendelt fájlszámot kapnak. Ezután olyan parancsokat használhatunk, amelyek ezzel a fájlszámmal működnek, olvasni vagy írni képesek vele az adott portot. A soros port lezárása a CLOSE paranccsal történik.

Néhány példa:

```
OPEN "COM1:4800" AS #5      \ open the first serial port with a speed of 4800 baud
PRINT #5, "Hello"           \ send the string "Hello" out of the serial port
dat$ = INPUT$(20, #5)       \ get up to 20 characters from the serial port
CLOSE #5                    \ close the serial port
```

Az OPEN parancs

A soros port megnyitása a következő módon történik:

```
OPEN comspec$ AS #fnbr
```

A 'fnbr' a használt fájlszám, értéke 1-10 között lehet. A # karakter opcionális.

A 'comspec\$' tartalmazza a kommunikáció jellemzőit. Ez egy füzér (lehet egy füzérváltozó) megadva a megnyitott sorosport paramétereit. Az alapértelmezett jellemzők: 9600 baud, 8 adatbit, paritás nincs, egy stop bit.

Formája "COMn: baud, buf, int, intlevel, DE, 9BIT, INV, OC, S2" ahol:

- 'n' a soros port száma COM1: vagy COM2:
- 'baud' a sebesség – ld. a későbbi Baud Rate táblázatot, ahol a sebességkorlátok szerepelnek. Alapértelmezett: 9600.
- 'buf' a vevő tároló mérete bájtokban (alapértelmezett mérete: 256). Az adó puffer mérete fixen 256 bájt.
- 'int' a felhasználó által definiált szubrutin, amit végre kell hajtani, ha a soros portra adat érkezett. Az alapértelmezés szerint nincs megszakítás.
- 'intlevel' a karakterek száma, amit meg kell várni a vevő pufferben, mielőtt a megszakítási rutint meghív-nánk. Az alapértelmezett érték 1 karakter.

Minden paraméter, kivéve a soros port nevét (COMn :) opcionális. Ha valamelyik paraméter kimarad, akkor a rákövetkező paraméterek is kimaradnak, és az alapértékeket fogják megkapni.

Öt opció adható meg a 'comspec\$' füzér végéhez. Ezek: DE, 9BIT, INV, OC és S2:

- 'DE' engedélyezi adatkimenet engedélyezve (Data output Enable (EN)) jelet az RS485 kommunikáció-nál. Ld. az "IEEE 485" alfejezetet a részletekért.
- '9BIT' jelzi, hogy az adás és vétel 9 bites. Ld. az "IEEE 485" alfejezetet a részletekért
- 'INV' specifikálja, hogy az adási és vételi jel polaritások invertáltak-e (de csak COM1: esetén).
- 'OC' specifikálja, hogy az adóláb (és DE a COM1:) nyitott kollektoros. Ez az opció COM1: és COM2: esetén is használható. Alapértelmezett a normál (0 - 3.3V) kimenet.
- 'S2' két STOP bit küldését specifikálja minden karakter adásának a végét (csak COM1:).

I/O lábak kiosztása

Ha egy soros portot megnyitunk, a szükséges port lábak automatikusan beállnak bemenetnek és kimenetnek, továbbá a SETPIN és PIN parancsok használhatatlanná válnak ezeken a lábakon. Ha a portot bezárjuk a CLOSE paranccsal, minden soros port által használt láb visszaáll konfigurálatlan állapotra, és a SETPIN paranccsal konfigurálhatókká válnak.

Az egyes COM portok kivezetései a kézikönyv elején lévő lábkiosztásnál láthatók. Ne feledjük, hogy Tx kimenet, és az Rx bemenet a Micromite-on. Az engedélyező láb (EN jel RS485-nél) is kimenet.

A jel polaritása szabványos azoknál az eszközöknél, amelyek TTL feszültséget használnak (RS232 feszültségeket lásd alább). Alapállapotban a feszültség magas, a start bit feszültsége alacsony, az adatbiteknél magas feszültség logikai 1, a stop bit feszültsége magas. Ezekkel a jelszintekkel közvetlenül kapcsolódhatunk olyan készülékek-hez, mint pl. a GPS modul (amely általában TTL feszültségszinteket használ).

Baud Rate

A 28 és 44-lábú Micromite COM1: portja a mikrokontroller UART perifériája, míg COM2: portot szoftverrel valósítjuk meg, és ezért nem annyira gyors. A maximális sebességet mindkét COM portnál korlátozza a CPU órajel frekvenciája (a processzor sebességét lehet változtatni a CPU SPEED paranccsal):

CPU sebesség	COM1: Maximum	COM2: Maximum
48 MHz	282000	19200
40 MHz (alapértelmezett)	230400	19200
30 MHz	115200	9600
20 MHz	115200	9600
10 MHz	57600	4800
5 MHz	38400	2400

Megjegyezzük, ezek a korlátok mellett bármilyen baud rate választható, például az 1111 bps érvényes sebesség mindkét portnál.

Példák

Soros port megnyitása alapértelmezett paraméterekkel:

```
OPEN "COM2:" AS #2
```

Soros port megnyitása a baud rate megadásával (4800 bit/sec):

```
OPEN "COM2:4800" AS #1
```

Soros port megnyitása a baud rate megadásával (9600 bit/sec), és 1KB vevő pufferrel:

```
OPEN "COM1:9600, 1024" AS #8
```

Mint az előző, de két stop bittel:

```
OPEN "COM1:9600, 1024, S2" AS #8
```

Példa minden beállításra, megszakítás, annak szintje, invertált jel, és két stop bit:

```
OPEN "COM1:19200, 1024, ComIntLabel, 256, INV, S2" AS #5
```

Olvasás és írás

Miután egy soros portot megnyitottuk, bármilyen fájlszámot használó paranccsal vagy funkcióval írni és olvasni lehet a portot. Általában a PRINT parancs a legjobb módszer az adatok elküldésére és az INPUT\$() függvény a legkényelmesebb módja az adatok vételének. Amikor az INPUT\$() függvényt használjuk a paraméterként megadott maximális számú karakterrel tér vissza, de lehet kevesebb, ha kevesebb karakter van a vételi pufferben. Valójában az INPUT\$() függvény azonnal visszatér egy üres füzérrel, ha nincs karakter a vételi pufferben.

A LOC() függvény nagyon hasznos, mert visszaadja a vevő pufferbe érkezett karakterek számát, amit utána az INPUT\$() függvénnyel kiolvashatunk. Az EOF() függvény visszatérési értéke igaz, ha nincs karakter a pufferben. A LOF() függvény visszaadja az adó pufferben fennmaradó üres karakterhelyek számát.

Mikor adunk a soros porton (azaz, PRINT #n, dat) a parancs várakozni fog addig, amíg a kimeneti puffer tele van, és megvárja, amíg lesz elegendő hely az új adatok számára. Ha a vételi puffer túlsordul a beérkező adatokkal, a soros port automatikusan eldobja a régebbi adatokat, hogy helye legyen az új adatoknak.

Soros portot bezárhatjuk a CLOSE paranccsal. Ez eldobja a bufferekben várakozó összes karaktert, felszabadítja a használt memóriát, megszünteti a megszakítást, ha be volt állítva, és a port által használt összes lábat nem konfigurált állapotba állítja vissza. Egy megnyitott soros port automatikusan bezáródik a RUN és a NEW parancsok hatására.

Megszakítások

A megszakítás szubrutin (ha megadtuk) úgy fog működni, mint egy külső I/O lábat használó általános megszakítás (lásd a "Az I/O lábak használata" részt).

Amikor a megszakításokat használunk, akkor tisztában kell lenniük azzal, hogy némi időt vesz igénybe a MMBasic-nek, hogy válaszoljon a megszakításra, és több karakter érkezik meg közben, különösen nagy adatátviteli sebességnél. Így például, ha megadtuk, hogy a megszakítás 200 karakternél történjen és a puffer 256 karakteres, akkor nagyon könnyen túlsordulhat a puffer, mire a megszakítás szubrutin ki tudja olvasni az adatokat. Ebben az esetben a puffert meg kell növelni 512 karakter méretűre vagy annál nagyobbra.

További soros portok

További soros portok adhatók a Micromite-hoz, felhasználva a SerialTx és SerialRx beágyazott C modulokat. Ezek az „Embedded C Modules” nevű alkönyvtárban találhatók, a Micromite förmver zip fájlban.

IEEE 485

A "DE" opció az OPEN comspec\$ COM1: parancsban meghatározza, hogy az adatok kimeneti engedélyezése (ENABLE) jelet generálja az IEEE 485 protokoll számára. Ez a jel jelenik meg a 28-lábú tok 7. lábán és általában magas szintű. Mielőtt egy bájtot továbbítunk a kimenet alacsony szintű lesz, és ha a bájt kivétele megtörtént, a kimenet ismét magas szintű lesz. Megjegyzendő, hogy a polaritása ellentétes a Maximite családnál megismerttel, és általában egy inverterre van szükséges az IEEE 485 adó-vevő DE bemenetének meghajtására.

Sok IEEE 485 rendszer használ 9 bites adatokat az adásnál és vételnél. A 9. bit jelöli, hogy a küldött vagy fogadott bájt címet vagy adatot tartalmaz. Ezt a "9BIT" opcióval állíthatjuk be az OPEN comspec\$ COM1: parancssal. Ezzel az opcióval az összes adatot bájt páronként kell küldeni a pár byte - az első bájt a 9. bit, a második bájt pedig a 8 bites adatokat. Az első bájtban vagy az ASCII '1' kell lenni, ami jelzi, hogy a 9. bit kell állítani, vagy "0"-át, ha nem akarjuk a 9. bitet 1-be állítani. Ezt a 9. bitet ezután hozzákapcsoljuk a második bájtához és bájt páros együtt képviseli a 9 bites adatküldést.

A következő kódrészlet három 9 bites adatot küld el. Az első a cím (9.bit=1), és a következő kettő az adat (9.bit=0):

```
OPEN "COM1: 4800, 9BIT" as #1
PRINT "1" + CHR$(211);
PRINT "0" + CHR$(23);
PRINT "0" + CHR$(0);
```

Megjegyezzük, hogy a PRINT parancsok végén szereplő „;” az automatikus CR/LF karakterek küldését megakadályozza.

Fogadott adatok felépítése hasonló. A 9 bites adatok két karakterként jelennek meg - az első az ASCII '1' vagy '0' jelzi a kapott adat 9. bitjét a második karakter pedig a másik 8 bit. Ez azt jelenti, hogy a BASIC programnak adatkárokat kell olvasni, és meghatározni 9. bitet (az első karakter), majd végrehajtani a megfelelő műveleteket a második karakterrel. Például:

```
IF LOC(#1) >= 2 THEN      ' check that we have at least two bytes
  A$ = INPUT$(1, #1) : B$ = INPUT$(1, #1)
  IF A$ = "1" THEN
    ' B$ contains an address
  ELSE
    ' B$ contains some data
  ENDIF
ENDIF
```

Az MMBasic nem ellenőrzi, hogy a COM porton párban nyomtatjuk vagy olvassuk-e az adatokat. Ha a program véletlenül elküld, vagy olvas csupán egyetlen karaktert, az meg fog zavarni minden ezt követő kommunikációt. Ne feledjük, hogy 9 bites módban az adó és vevő pufferek mérete ténylegesen a felére csökken, mert mindegyik 9 bites érték két bájtban tárolódik.

Olcsó RS-232 illesztés

A modemek, a PC vezetékes soros portja, vizsgálati eszközök, stb mindegyike az RS-232 jelátviteli rendszert használja. Ez pontosan ugyanaz, mint a Micromite soros TTL átviteli rendszere két különbséggel:

- A feszültség szintek RS-232-nél +12V és -12V míg a TTL soros +3.3V és 0V feszültséget használ.
- A jelek invertáltak, vagyis a +3,3 volt -12V míg 0V-nak +12V felel meg.

Vásárolható olcsó RS-232 TTL átalakító, de az lenne praktikus, ha a Micromite közvetlenül tudna kapcsolódni az RS-232 interfészhez.

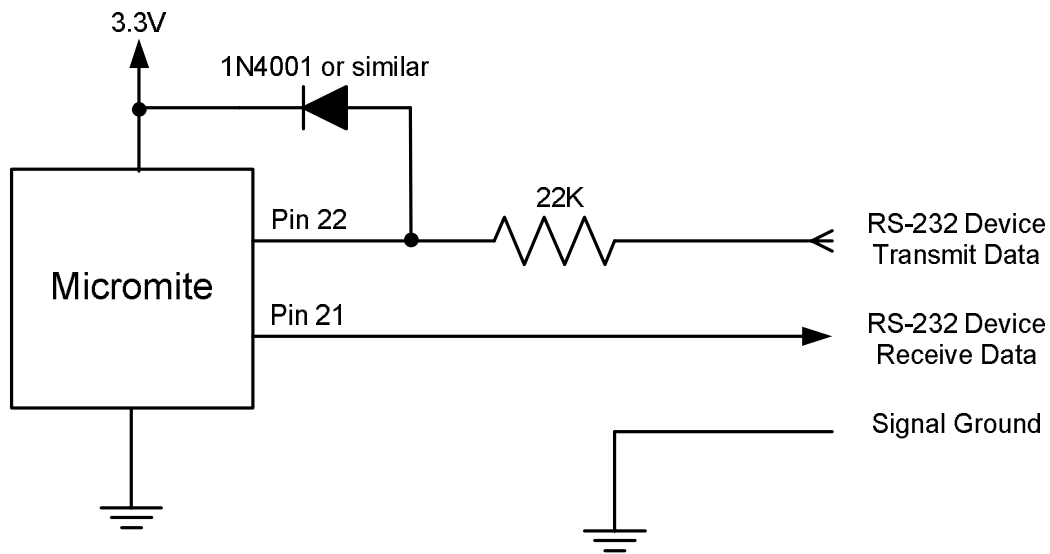
Az első probléma az, hogy a feszültség szintek fordítottak. A Micromite COM1-nél: megadható, hogy fordítsa meg az adó- és a vevő jelének a polaritását (az "INV" opció), így ez könnyen megoldható.

Az adatok fogadásakor (vagyis ha ± 12 V-os jelet küld a távoli RS-232 berendezés) könnyű a feszültséget korlátozni egy soros ellenállással, (mondjuk) 22k Ω és egy dióda, amely megfogja a pozitív feszültség 3.3 V-nál. A Micromite bemeneti impedanciája nagyon magas, ezért az ellenállás nem okoz feszültség esést, de ez nem jelenti azt, hogy ha a jel eléri a maximum +12V-ot a dióda kinyit, és amikor eléri a -12V-ot, akkor a Micromite lábán lévő belső védődióda akadályozza meg a negatív feszültség bejutását a tokba.

Az adójel (a Micromite-től az RS-232 eszköz felé) közvetlenül csatlakozhat a távoli eszköz bemenetére. A Micromite által küldött feszültség 0V és 3.3V értékűek, de a legtöbb RS-232 bemeneti küszöb +1V körüli így a Micromite jele értelmezhető.

Ezek a megoldások az RS-232 jelzési szintjeitől eltérnek, de 1-2 méteres távolságokon remekül működnek.

Összegezve, ezt az áramkört használjuk (28-lábú toknál):



és nyissuk meg a COM1: portot inverz opcióval.

```
OPEN "COM1: 4800, INV" AS #1
```

Ne feledjük, hogy a külső védő dióda elhagyható olyan I/O láb esetén, amelyik nem 5V toleráns, mert ott már be van építve egy ilyen dióda.

I2C kommunikáció – B melléklet

Az Inter Integrated Circuit (I²C) busz a Philips (most NXP) fejlesztette ki, integrált áramkörök közötti adatátvitelre. Ezt az implementációt Gerard Sexton írta, és az I²C szabványos definícióit: http://www.nxp.com/documents/user_manual/UM10204.pdf dokumentum tartalmazza.

Az I²C mester módjánál négy parancsot használhatunk:

I2C OPEN speed, timeout [,PU]	<p>Az I²C modul mester módját engedélyezzük.</p> <p>‘speed’ egy érték 10 és 400 között (buszsebesség 10 kHz - 400 kHz).</p> <p>‘timeout’ olyan érték ezredmásodpercben, ami után a mester által küldött és fogadott parancsai megszakadnak, ha még nem fejeződtek be. A minimális érték 100. A nulla érték letiltja az időtúllépés kezelését (bár ez nem ajánlott).</p> <p>‘PU’ (ha megadjuk) engedélyezi a gyenge (100K) felhúzó ellenállásokat az órajel és az adatkivezetéseken. Az I²C modul szokásos használata egy kisebb, tipikusan 10K felhúzó ellenállást igényel, de kis sebességnél és rövid jelvezetékeknél ez elhagyható.</p>
I2C WRITE addr, option, sendlen, senddata [,senddata]	<p>Adatküldés az I²C szolga eszközbe.</p> <p>‘addr’ az I²C szolga címe.</p> <p>‘option’ egy szám 0 és 3 között (normál esetben 0-ra van állítva)</p> <ul style="list-style-type: none">1 = megtartja a buszvezérlést a parancs után (a STOP állapot nem lesz elküldve a parancs befejezésekor)2 = a címet 10 bites címként kezeljük3 = 1 és 2 kombinálva (tartsuk a buszt, és 10 bites címezést használjunk). <p>‘sendlen’ a küldött bájtok száma.</p> <p>‘senddata’ a küldött adatok – ezt több féle módon adhatjuk meg (minden küldött adat 0 és 255 között van):</p> <ul style="list-style-type: none">• A parancsnál szereplő bájtokat egyenként kell megadni. Például: I2C WRITE &H6F, 1, 3, &H23, &H43, &H25• Az adatok lehetnek egydimenziós tömbben, amit üres zárójellel jelzünk. Az adatokat az első elemtől kezdődően írjuk. Például: I2C WRITE &H6F, 1, 3, ARRAY()• Az adat lehet füzér változó, de nem állandó. Például: I2C WRITE &H6F, 1, 3, STRING\$
I2C READ addr, option, rcvlen, rcvbuf	<p>Adatfogadás az I²C szolga eszköztől.</p> <p>‘addr’ az I²C szolga címe.</p> <p>‘option’ egy szám 0 és 3 között (normál esetben 0).</p> <ul style="list-style-type: none">1 = a buszvezérlést megtartja a parancs után (a stop feltétel nem lesz elküldve a parancs végén)2 = a címet 10 bites címként kezeljük3 = 1 és 2 kombinálva (tartsuk a buszt, és 10 bites címezést használjunk). <p>‘rcvlen’ a vett bájtok száma.</p> <p>‘rcvbuf’ egy puffér az adatok vételére – ez lehet egy füzérváltozó (pl.: t\$), vagy egydimenziós, számokat tartalmazó tömb, a dimenzió megadása nélkül (pl.: data()) vagy egy normál numerikusváltozó (ebben az esetben rcvlen értékének 1-nek kell lenni).</p>
I2C CLOSE	<p>Tiltja a mester I²C modult és visszatér, a használt lábakat inaktívvá állítva. Ezután a lábak a SETPIN paranccsal konfigurálhatók. Ez a parancs STOP állapotot is kiküld, ha busz még aktív.</p>

Hasonlóan itt a négy parancs a szolga módnál:

I2C SLAVE OPEN addr, mask, option, send_int, rcv_int	<p>Az I²C modul szolga módját engedélyezzük.</p> <p>‘addr’ az I²C címe a szolgának.</p> <p>‘mask’ a cím maszk (normál esetben 0, ha a bitek 1-be vannak mindig egyezés van). Ez lehetővé teszi, hogy a szolga többszörös címekre is válaszoljon.</p> <p>‘option’ egy szám 0 és 3 között (normál esetben 0).</p> <p>1 = megengedi, hogy az MMBasic válaszoljon egy általános hívási címre. Ha ez történik, MM.I2C értéke 4 lesz.</p> <p>2 = a cím 10 bitesként lesz kezelve</p> <p>3 = 1 és 2 kombinációja</p> <p>‘send_int’ a meghívandó szubrutin, mikor a modul érzékeli, hogy a mester adatot vár.</p> <p>‘rcv_int’ meghívandó szubrutin, mikor a modul adatot kap a mestertől.</p> <p>Megjegyezzük, ezt az első beérkezett bájt aktivizálja, így a program várakozhat, amíg az összes adat vétele megtörtént.</p>
I2C SLAVE WRITE sendlen, senddata [,senddata]	<p>Adatot küld az I²C mesternek. Ezt a parancsot kell használni a küldési megszakításban (vagyis a "send_int" szubrutinban, amikor a mester adatokat vár). Alternatívaként egy jelzőbitet lehet beállítani a megszakítás szubrutinban, és a fő program hurokban ezt figyelve hajtjuk végre ezt a parancsot.</p> <p>‘sendlen’ a küldött bájtok száma.</p> <p>‘senddata’ a küldött adatok. Ez többféle módon megadható részletek az I2C WRITE parancs leírásában.</p>
I2C SLAVE READ rcvlen, rcvbuf, rcvd	<p>Adatokat fogad az I2C master eszköztől. Ezt a parancsot kell használni a vételi megszakításakor (vagyis az "rcv_int" szubrutinban, amikor a mester küldött néhány adatot). Alternatívaként egy jelzőbitet is be lehet állítani a vételi megszakítás szubrutinban és ennek a főprogram hurokban történő vizsgálatával hívjuk meg a parancsot, amikor a jelzőbit 1-be be van állítva.</p> <p>‘rcvlen’ a vett bájtok maximális száma.</p> <p>‘rcvbuf’ egy tároló (puffer) az adatok vételére –ez lehet egy változó függér(pl.: t\$), vagy számokból álló egy dimenziós tömp, dimenzió nélkül megadva (pl.: data()), vagy egy számváltozó (ekkor rcvlen értékének 1-nek kell lenni).</p> <p>‘rcvd’ a parancs által vett bájtok aktuális számát tartalmazza.</p>
I2C SLAVE CLOSE	<p>Tiltja a szolga I2C modult és visszatér, a használt lábakat inaktívra állítja. Ezután a lábak a SETPIN paranccsal konfigurálhatók</p>

Az I²C írás vagy olvasás parancsokat követően, a belső MM.I2C változó be lesz állítva, a művelet eredményétől függően:

- 0 = A parancs hiba nélkül befejeződött.
- 1 = Egy NACK válasz érkezett
- 2 = A parancs időtúllépése

A régebbi MMBasic-et használóknak

Az I²C protokoll jelen megvalósítása általában kompatibilis az előző verziókban lévőkkel. A különbségek:

- Az átnevezett parancsok működése nem változott: I2CEN most I2C OPEN, I2CSEND most I2C WRITE, I2CRCV most I2C READ és I2CDIS neve most I2C CLOSE. Hasonlóan, I2CSEN most I2C SLAVE WRITE, stb.
- Mester megszakítás nem támogatott.
- A NUM2BYTE parancs és a BYTE2NUM () függvény megszűnt (használjuk helyette a PEEK függvényt és a POKE parancsot).

További I2C portok

További I²C mester portok adhatók a Micromite-hoz, felhasználva az I2Cport beágyazott C modult. Ez az „Embedded C Modules” nevű alkönyvtárban található, a Micromite főmver zip fájlban.

7 és 8 bites címzés

A parancsokban használt szabványos címek 7 bitesek, nyolcadik, read/write bit nélkül. Ezt a bitet az MMBasic automatikusan kezeli, és hozzáadja a címhez.

Néhány gyártó 8-bites címeket biztosít, amelyek a read/write bitet tartalmazzák. Ezt felismerhetjük, mert külön szerepeltet egy írási, illetve egy olvasási címet. Ilyenkor címként a felső hét bitet használjuk

Például: Ha az olvasási cím 9B (hexadecimális) és az írási cím 9A (hexadecimális), akkor kizárólag az első hét bitet adja a címet, és ez 4D (hexadecimális). Egyszerűbb a meghatározás, ha az írási címet 2-vel elosztjuk.

A címtartomány ellenőrzésére a gyártó 8 bites címzést használ 7 bites helyett. Minden 7 bites címnek a 08-77 (hex) tartományban kell lenni. Ha a szolgálja címe nagyobb, mint ez a tartomány, a gyártó 8 bites címet ad meg.

10 bites címzés

A 10 bites címzést úgy tervezték, hogy kompatibilis legyen a 7 bites címzéssel, amely lehetővé teszi, hogy a fejlesztők, keverhessék a mindkét típushoz tartozó eszközöket egyetlen buszon. A 10 bites címeket használó eszközöket az adatlapon a gyártó egyértelműen jelzi.

10-bites címzésnél a szolgálja címet két bájtban küldjük el. Az első bájtban egy speciális bitminta jelzi, hogy 10 bites címről van szó. Ezt a küldést az MMBasic automatikusan kezeli, amikor a 'option' paraméterben ez 10 bites címzést állítunk be. 10 bites címek 0-3FF hex tartományban lehet.

Mester/Szolgálja módok

A mester és szolgálja módban bekapcsolható egyszerre. Azonban, ha egy mester parancsa folyamatban van, a szolgálja funkció lesz "idle", felfüggesztett lesz, amíg a mester el nem engedi a buszt. Hasonlóképpen, ha egy szolgálja parancs van folyamatban, a mester parancsok nem lesz elérhetők, amíg a szolgálja-tranzakció be nem fejeződik.

Master módban, az I²C küldés és fogadás parancsai után a program nem folytatódik, amíg a parancs be nem fejeződik, vagy időtúllépés nem történik (ha az időtúllépés meg lett adva).

A szolgálja mód MMBasic megszakítást használ, hogy jelezze a helyzet megváltozását, és a Micromite-nak ebben a rutinban kell írni/olvasni az adatokat I²C mesterbe/ből. Ez ugyanúgy működik, mint egy külső I/O láb által okozott általános megszakítás.

I/O lábak

A kézikönyv elején található lábkiosztásoknál szerepelnek az I²C adatvonalához (SDA) és a órajelhez (SCL) tartozó lábszámok.

A lábak mindegyikét fel kell húzni egy beiktatott külső ellenállással (tipikus értéke 10kΩ 100kHz-nél illetve 2kΩ 400 kHz-nél). Gyenge (100K) felhúzó ellenállásokat kapcsolhatunk be az órajel és az adatkivezetéseken az I2C OPEN parancsban. Az I²C modul szokásos használata egy kisebb, tipikusan 10K felhúzó ellenállást igényel, de kis sebességnél és rövid jelvezetékeknél ez elhagyható.

Ha az I²C CLOSE parancsot használjuk, a használt I/O lábak nem konfigurált állapotba kerülnek, és a SETPIN paranccsal ismét konfigurálhatók.

Ha az I²C busz sebessége 150 kHz felett van, a vezetékezés is lényegessé válik. Ideális esetben a kábelnek olyan rövidnek kell lenni, amennyire lehetséges (kapacitás csökkentése), és az adat és órajel vezetékek ne egymáshoz közel, párhuzamosan fussanak, hanem legyen közöttük földvezeték (áthallás csökkentése).

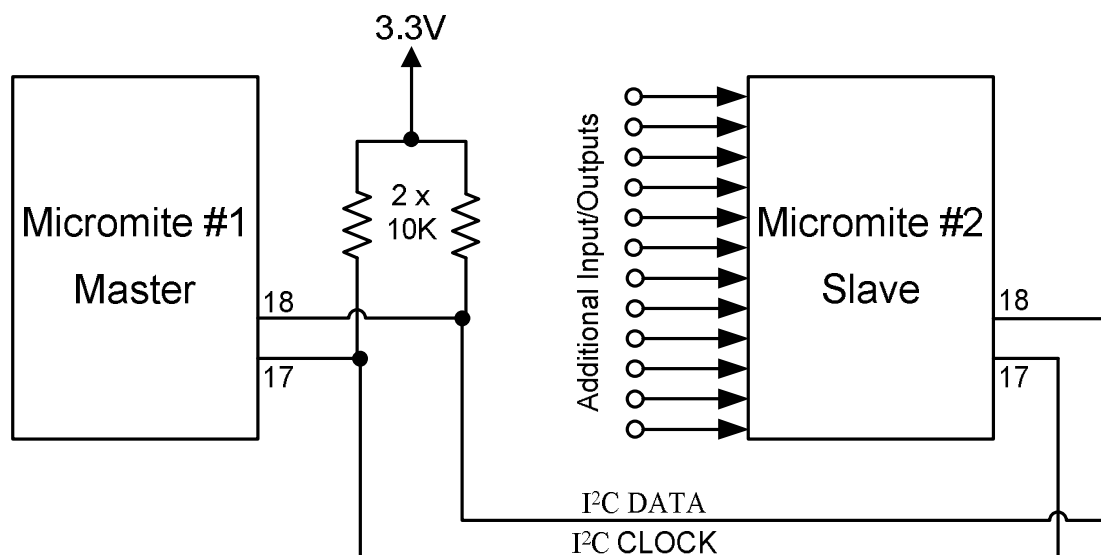
Ha az adatvonal nem stabil, és az órajel magas, vagy zajos, akkor az I²C perifériák "összezavarodhatnak" és a busz kifagy (CLOCK vonal alacsony szinten marad). Ha nem kell nagy sebesség, akkor használjuk a biztonságos, 100 kHz-es buszsebességet.

Példa

Az I²C ideális integrált áramkörök közötti kommunikációra. Példaként nézzük, hogy mikor nincs elég soros port, vagy I/O vagy valami egyéb a Micromite-ban az adott alkalmazáshoz. Ebben az esetben egy második Micromite-ot lehet szolgálként használni bővítésre.

A példában a második Micromite egy általános célú I/O bővítő tokká válik 17 I/O lábbal, és azok bármelyike a mester által dinamikusan konfigurálható, mint az analóg bemenet, vagy digitális bemenet/kimenet. A rutinokat a mesterben egyszerű használni (SETPIN a szolgálja I/O a beállítása SPIN () az irány beállítása, és a mesterben futó programnak nem kell tudnia, a másik tok I/O lábainak fizikai elhelyezkedéséről. Minden kommunikáció az I²C buszon keresztül történik.

A következő ábra mutatja két 28-lábú tok összekötését:



A szolgában futó program:

A szolgában először fel kell élednie az I²C interfészének, hogy tudjon válaszolni, a mester kérésére. Hogy ez biztosított legyen, egy végtelen hurokban kell várakoznia, hogy a mester kérését az I²C megszakításban le tudja kezelni.

A lenti programban a szolga hallgat a 26 (hex) I²C címen, a mester által küldött hárombájtos parancsra. Az üzenet formátuma a következő:

- Első bájtt a parancs típusa. Ez három érték valamelyike lehet; 1 – a lábat konfiguráljuk, 2-azt jelenti, hogy a láb kimenet lesz, 3-as láb bemenet lesz.
- Második bájtt a konfigurálandó láb száma.
- Harmadik bájtt a konfigurációs számot tartalmazza (ha a parancs bájtt értéke 1), a láb kimeneti állapota (ha a parancs bájtt értéke 2) vagy nem érdekes (ha a parancs bájtt 3).

A konfigurációs szám, amivel beállíthatunk egy szolga I/O lábat, ugyanaz, mint amit a korábbi Maximite MMBasic verziókban (a SETPIN parancsban) használtunk.

- | | |
|---|---|
| 0 | Nem konfigurált vagy inaktív |
| 1 | Analóg bemenet |
| 2 | Digitális bemenet |
| 3 | Frekvencia bemenet |
| 4 | Periódus bemenet |
| 5 | Számláló bemenet |
| 8 | Digitális kimenet |
| 9 | Nyitott kollektoros digitális kimenet. Ebben a módban az SPIN() parancs a kimeneten lévő értéket adja vissza. |

A mester által küldött olyan parancs esetén, amely egy bemenetet kér, a mesternek kell majd kiadni egy második I²C utasítást 12 bájtt olvasására. A szolga a 12 karakterből álló füzér elküldésével válaszol.

A program összeomlik, ha a mester hibás parancsot küld! Például akkor, ha megpróbálunk olvasni egy nem bemeneti lábról. Ha ez bekövetkezik, hiba történik, és az MMBasic kilép a parancssorba.

Ahelyett, hogy az összes, mester által okozott lehetséges hibát kezelnénk, a program a watchdog időzítőt használja. Ha hiba történik, akkor a watchdog egyszerűen újraindítja a Micromite-t és a program újra indul (mert az AUTORUN be van kapcsolva), és várja a következő üzenetet a mestertől. A mester ilyenkor kiírhatja, hogy valami nem stimmel, mert letelt az időkorlát. Itt a szolgán futó teljes program:

```
OPTION AUTORUN ON
DIM msg(2)
I2C SLAVE OPEN &H26, 0, 0, WriteD, ReadD
```

' tomb az üzenet tarolasara
' szolga címe, ez 26 (hex)

```

DO                                ' vegtelen ciklus
    WATCHDOG 1000                ' így lehet hibából feleledni
LOOP

SUB ReadD                        ' üzenet vetele
    I2C SLAVE READ 3, msg(), recvd ' az üzenet a tombbe kerül
    IF msg(0) = 1 THEN           ' parancs = 1
        SETPIN msg(1), msg(2)    ' I/O láb konfigurálása
    ELSEIF msg(0) = 2 THEN       ' parancs = 2
        PIN(msg(1)) = msg(2)     ' I/O láb kimenet
    ELSE                         ' a parancs csak 3 lehet
        s$ = str$(pin(msg(1))) + Space$(12) ' az I/O láb bemenet, beolvasas
    ENDIF
END SUB                          ' megszakítás vége

SUB Writed                       ' keres a mestertől
    I2C SLAVE WRITE 12, s$       ' utolsó merés elküldése
END SUB                         ' megszakítás vége

```

A mesterben lévő illesztő rutinok:

Ezeket a rutinokat egy másik Micromite vagy Maximite eszközön, vagy más számítógépen is lehet futtatni, aminek I²C interfésze van. Azt feltételezi, hogy a szolga Micromite a 26 (hexadecimális) I²C címen figyel.

Ha szükséges, könnyen megoldható, hogy különböző címeken több Micromite-ot érjünk el melyek bővítőként működnek. Számuk szinte korlátlanul bővíthető!

A mester két alprogramot és egy függvényt használ a szolga vezérlésére:

SSETPIN pin, cfg	Ez a szubrutin beállít egy I/O lábat a szolgában. Úgy működik, mint az MMBasic SETPIN parancsa, 'cfg' lehetséges értékeit korábban már leírtuk.
SPIN pin, output	Ez a szubrutin állítja be a szolgában a kimenetet (magas, vagy alacsony).
nn = SPIN(pin)	A függvény a szolgában lévő bemenet állapotát olvassa be a lábról.

Például a szolgában a 3-as lábra kötött feszültség kiírása:

```

SSETPIN 3, 1
PRINT SPIN(3)

```

Másik példa: LED villogtatás, amelyik a szolga 15. lábára van kötve:

```

SSETPIN 15, 8
SPIN 15, 1
PAUSE 300
SPIN 15, 0

```

Az említett három rutin:

```

' a szolgában az I/O láb konfigurálása
SUB SSETPIN pinnbr, cfg
    I2C OPEN 100, 1000
    I2C WRITE &H26, 0, 3, 1, pinnbr, cfg
    IF MM.I2C THEN ERROR "Szolga nem valaszol"
    I2C CLOSE
END SUB

' szolgában lévő I/O láb konfigurálása kimenetnek
SUB SPIN pinnbr, dat
    I2C OPEN 100, 1000
    I2C WRITE &H26, 0, 3, 2, pinnbr, dat
    IF MM.I2C THEN ERROR "Szolga nem valaszol"
    I2C CLOSE

```

```
END SUB
```

```
' szolgában lévő bemeneti I/O láb állapotának olvasása
FUNCTION SPIN(pinnbr)
  LOCAL t$
  I2C OPEN 100, 1000
  I2C WRITE &H26, 0, 3, 3, pinnbr, 0
  I2C READ &H26, 0, 12, t$
  IF MM.I2C THEN ERROR "Szolga nem valaszol"
  I2C CLOSE
  SPin = VAL(t$)
END FUNCTION
```

Egyvezetékes kommunikáció – C melléklet

Dallas Semiconductor (MAXIM) által kifejlesztett egyvezetékes (1-wire) protokoll egyetlen jelvezetéken tud kommunikálni integrált áramkörökkel. Ennek a használatát MMBasic-re Gerard Sexton írta meg.

Mindössze három parancsot használhatunk:

ONEWIRE RESET pin	Reszteli az 1-Wire buszt
ONEWIRE WRITE pin, flag, length, data [, data...]	Adott számú bájtot küld
ONEWIRE READ pin, flag, length, data [, data...]	Adott számú bájtot fogad

Ahol:

pin - A használt Micromite I/O láb. Ez bármelyik láb lehet, amely digitális I/O-ra alkalmas.

flag - A következő opciók kombinációja:

1 – Resztet küld parancs előtt

2 - Resztet küld parancs után

4 – Csak egy bitet küld/fogad egy adatbájt helyett

8 - Legyen egy erős felhúzás a lábon (a láb magasra lesz állítva, és a open drain tiltva lesz)

length – A küldött vagy vett adatok száma

data – Adat, amit küldünk, vagy változó, amiben fogadjuk az adatot.

Az adatelemek számának meg kell egyeznie a length paraméter értékével.

És egy automatikus változó

MM.ONEWIRE

Igaz, ha az 1-wire eszköz létezik

Fontos: a CPU sebessége minimum 10 MHz legyen.

A végrehajtott parancs után, az I/O láb nem konfigurált állapotba kerül, kivéve, ha a 8-as értékű 'flag'-et használunk.

Ha resztelünk, az MM.ONEWIRE automatikus változó igaz lesz, amikor az eszközt megtalálta a Micromite. Ez létrejön az ONEWIRE RESET parancs használatakor, és az ONEWIRE READ és ONEWIRE WRITE parancsoknál is a reszt után (flag = 1 vagy 2).

Az MMBasic-et használó régebbi eszközöknél

Az egyvezetékes protokoll implementációja általában kompatibilis a régebbi verziókkal a következő különbségekkel:

- A parancsok most két szavasak ott, ahol korábban egy szavasak voltak. Például OWWRITE most ONEWIRE WRITE.
- Nem használhat tömb vagy karakterfüzér változót a 'data' paraméter. Egy vagy több numerikus változót kell használni.
- A reszt parancs (ONEWIRE RESET) nem fogadja el a 'presence' változót (használjuk helyette a MM.ONEWIRE változót).
- Az OWSEARCH parancs és a OWCRC8 () és OWCRC16 () függvények nincsenek megvalósítva.

Az 1-Wire protokollt gyakran használják a DS18B20 hőmérsékletmérő szenzorral való kommunikációnál. Ezt segítő, az MMBasic tartalmazza a TEMPR() függvényt, mely lehetővé teszi, hogy közvetlenül olvassuk a hőmérsékletet egy DS18B20-as tokból, az 1-wire parancsok használata nélkül.

SPI kommunikáció – D melléklet

A Serial Peripheral Interface (SPI) kommunikációs protokoll integrált áramkörök közötti adatátvitelre használt. A Micromite mesterként működik (azaz mesterként generálja az órajelet).

I/O lábak

Az SPI OPEN parancs automatikusan konfigurálja a megfelelő I/O lábakat. Az SPI kivezetéseket a kézikönyv elején lévő lábkiosztásban megadtuk. MISO a Mester In Slave Out rövidítése és mivel a Micromite mindig a mester, ez láb bemenetnek van konfigurálva. Hasonlóképpen MOSI Mester Out Slave In rövidítése és ezért ez a láb kimenetre lesz állítva.

Amikor az SPI CLOSE parancsot használjuk, a lábak visszaállnak nem konfigurált állapotba. Ezek után a SET-PIN parancssal szabadon átkonfigurálhatjuk.

SPI Open

SPI használatához az SPI csatornát először meg kell nyitni.

Az SPI vagy SPI2 csatorna megnyitási parancsa:

```
SPI OPEN speed, mode, bits
```

Ahol:

- 'speed' az órajel sebessége. Ez egy szám, és Hz-ben kell megadni. Maximális értéke a CPU órajelének egy negyede (pl., 10000000 40Mhz-es CPU órajelnél).
- 'mode' egy szám, az átviteli módot kell megadni, az alábbi táblázat alapján (Transmission Format).
- 'bits' a küldött/fogadott bitek száma. Ez lehet 8, 16 vagy 32.
- a CS láb kezelése, ha szükséges, akkor a program feladata

Adatátviteli formátum

A legnagyobb helyiértékű bitet küldjük és fogadjuk először. Az átviteli formátum megadható a 'mode' üzemmóddal az alábbiak szerint. Mode 0 a legelterjedtebb formátum.

Mode	Leírás	CPOL	CPHA
0	Órajel aktív magas, az adatot fogadjuk annak felfutó élénél, kiküldjük a lefutó élénél.	0	0
1	Órajel aktív magas, az adatot fogadjuk annak lefutó élénél, kiküldjük a felfutó élénél.	0	1
2	Órajel aktív alacsony, az adatot fogadjuk annak lefutó élénél, kiküldjük a felfutó élénél.	1	0
3	Órajel aktív alacsony, az adatot fogadjuk annak felfutó élénél, kiküldjük a lefutó élénél.	1	1

A részletesebb leírást ld.: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Szabványos Küldés/Fogadás

Mikor az SPI csatorna nyitva van adatok küldhetők és fogadhatók az SPI vagy SPI2 funkciókkal. A formája:

```
received_data = SPI(data_to_send)
```

Vegyük figyelembe, hogy egyetlen SPI tranzakció elküldi az adatokat, miközben egyidejűleg adatot kap a szolgáltól. A "data_to_send" az elküldött adatokat tartalmazza, a függvény visszatérési értéke pedig a tranzakció során kapott adatokat tartalmazza. Ha nem szeretne elküldeni semmilyen adatot, azaz csak venni szeretne, akkor bármilyen számot (pl nulla) lehet használni az adatküldésre. Hasonlóképpen, ha nem akarjuk használni a kapott adatokat, akkor egy változóba írjuk és eldobjuk.

Adatfolyam Küldés/Fogadás

Adatokat folyamként is küldhetjük SPI vagy SPI2-vel:

```
SPI WRITE nbr, data1, data2, data3, ... etc
```

vagy

```
SPI WRITE nbr, string$
```

vagy

```
SPI WRITE nbr, array()
```

Az első módszernél 'nbr' a küldendő adatok száma, és az adatok sorban egymás után következnek (azaz 'data1', 'data2' stb).

A második és harmadik módszernél az adatokat "string \$" vagy "array ()" tartalmazza (ami, egydimenziós tömb egész vagy lebegőpontos számokkal). A fűzér hossza, vagy a tömb méretének azonosnak vagy nagyobbknak kell, mint a 'nbr' darabszám. A szolgától érkező adatokat eldobjuk.

Az adatokat folyamként is veheti az SPI vagy SPI2:

```
SPI READ nbr, array()
```

Ahol 'nbr' a vett adatok száma, és array() egydimenziós egész tömb, ahol az érkező adatokat elhelyezzük. A parancs nullákat küld, míg a szolgából olvasunk.

SPI Close

Ha szükséges, az SPI vagy SPI2 kommunikációs csatornát le lehet zárni a következő módon (az I/O láb inaktíva lesz állítva):

```
SPI CLOSE
```

További SPI portok

További SPI portok adhatók a Micromite-hoz, felhasználva az SPIPortt beágyazott C modult. Ez az „Embedded C Modules” nevű alkönyvtárban található, a Micromite főmver zip fájlban.

Példák

A következő példában egy parancsot 80 (hex) küldünk, és két bájtot kapunk a szolgál SPI eszköztől, felhasználva a szabványos küldés/fogadás funkciót::

```
PIN(10) = 1 : SETPIN 10, DOUT      ' 10-es labon lesz az engedelyezo jel
SPI OPEN 5000000, 3, 8              ' sebesseg 5MHz adathossz 8 bit
PIN(10) = 0                         ' beallitjuk az eng. labat(aktiv alacsony)
junk = SPI(&H80)                    ' kuldjuk a parancsot, valaszt eldobjuk
byte1 = SPI(0)                      ' elso bajt vetele a szolgatol
byte2 = SPI(0)                      ' masodik bajt
PIN(10) = 1                         ' szolga deaktiválása
SPI CLOSE                           ' vege, csatorna zarasa
```

A következő hasonló a fenti példához, de most az átvitel használja a tömeges (bulk) átviteli parancsokat:

```
OPTION BASE 1                      ' tombunk 1-től indexelt
DIM data%(2)                       ' tomb definíálása az adatok vetelehez
PIN(10) = 1 : SETPIN 10, DOUT      ' 10-es lábon lesz az engedelyezo jel
SPI OPEN 5000000, 3, 8              ' sebesseg 5MHz adathossz 8 bit
PIN(10) = 0                         ' beallitjuk az eng. labat(aktiv alacsony)
SPI WRITE 1, &H80                   ' parancskuldes
SPI READ 2, data%()                 ' ket bajt vetele a szolgatol
PIN(10) = 1                         ' szolga inaktívalása
SPI CLOSE                           ' csatorna zarasa
```

SPI használata színes LCD/TFT kijelzőknél és érintőképernyő kezelésnél

A Micromite-nak csak egy szabványos SPI csatornája van, így ezt meg kell osztani a kijelző, az érintőképernyő vezérlője (ha ezeket a szolgáltatásokat használjuk) és a BASIC program között. Ehhez az SPI portot BASIC-ben meg kell nyitni, majd lezárni anélkül, hogy a képernyőre adatot küldenénk, vagy az érintőképernyőről adatot fogadnánk. Tehát nem használhatunk olyan parancsokat, mint a CLS, LINE, stb. és a TOUCH () függvény.

Alább egy példát láthatunk. SPI port nyitása és zárása függvényen belül történik, a grafikus vagy érintési parancsok zavarása nélkül:

```

PIN(26) = 1
SETPIN 26, DOUT
CIRCLE 100, 100, 50
nbr% = ReadSPI%()
x% = TOUCH(X)
nbr2% = ReadSPI%()
y% = TOUCH(Y)
END

' SPI adat vevő függvény
FUNCTION ReadSPI%()
    SPI OPEN 4000000,0,8
    PIN(26) = 0
    SPI write 3,3,0,0
    SPI read 1, ReadSPI%
    PIN(26) = 1
    SPI CLOSE
END FUNCTION

' 1-re állitjuk a labat setup előtt
' 26-os lab az egngedelyezo jel
' kor rajzolasa(SPI port-ot használja)
' SPI adat vétele
' erintes X koordinata használja az SPI portot)
' SPI adat vetele masodszor
' érintes Y koordinátája(SPI portot használ)

' 4Mz sebesség, mod 0, 8 bit
' CS alacsony
' parancs küldése
' és a valasz kapasa
' CS magas

```

Micromite MMBasic 5.4 Quick Reference E melléklet

Micromite MMBasic Version 5.4 Quick Reference

Program Management

CONTINUE
CPU speed
CPU SLEEP [sec [, abortpin]]
CPU RESTART
CSUB name(type [, type]) rtype
END CSUB
CFUNCTION name type [,type] [AS type]
END CFUNCTION
DEFINEFONT #Nbr
END DEFINEFONT
EDIT
END
LIBRARY SAVE | DELETE | LIST
LIST [ALL]
MEMORY
NEW
POKE BYTE | WORD | VAR | VARTBL, addr, dat
RUN
TIMER = msec
TRACE ON | OFF | LIST nn
VAR SAVE var [, var]... | RESTORE | CLEAR
WATCHDOG timeout | OFF
XMODEM SEND | RECEIVE [filename\$]
nbr = PEEK(BYTE | WORD | VARADDR | CFUNADDR
| VAR | VARTBL | PROGMEM, args)

Input/Output

SETPIN pin, cfg [, option]
cfg = OFF | AIN | DIN | FIN | PIN | CIN | DOUT
option = PULLUP | PULLDOWN | OC | gate | cycles
SETPIN pin, OFF | INTH | INTL | INTB, target [, option]
option = PULLUP | PULLDOWN
PIN(pin) = value
PORT(start, nbr [,start, nbr]...) = value
PULSE pin, width
pulsewidth = PULSIN(pin, polarity [, t1 [, t2]])
value = PIN(pin)
value = PORT(start, nbr [,start, nbr]...)

Operators

NOT ^ Logical inverse, exponentiation
* / \ Multiply, division (float & integer)
MOD Modulus (remainder)
+ - Addition and subtraction
x << y x >> y Shift bits left/right by y bits
<> < > Not equals, less/greater than
<= >= Less/greater than or equals
AND OR XOR Logical and, or, exclusive or

Variables

Identifier = [A-Z] _ [A-Z] 0-9 [.] _] Max 32 chars.
Suffix: FLOAT = ! INTEGER = % STRING = \$
Literal Number = [&H | &O | &B] number
MM.VER MM.DEVICES\$
MM.ERRNO MM.ERRMSG\$
MM.HRES MM.VRES
MM.FONTHEIGHT MM.FONTWIDTH
MM.WATCHDOG
MM.I2C MM.ONEWIRE

GUI Controls (MM+)

OPTION CONTROLS nn
GUI AREA #ref, X, Y [, width, height]
GUI BUTTON #ref, caption\$, X, Y [, w, h, FC, BC]
GUI CAPTION #ref, text\$, X, Y [, just\$, FC, BC]
GUI CHECKBOX #ref, caption\$, X, Y [, size, colour]
GUI DISPLAYBOX #ref, X, Y [, width, height, FC, BC]
GUI FRAME #ref, caption\$, X, Y [, width, height, colour]
GUI LED #ref, caption\$, X, Y [, radius, colour]
GUI NUMBERBOX #ref, X, Y [, width, height, FC, BC]
GUI RADIO #ref, caption\$, X, Y [, radius, colour]
GUI SPINBOX #ref, X, Y, w, h [, FC, BC, Step, Min, Max]
GUI SWITCH #ref, caption\$, X, Y [, width, height, FC, BC]
GUI TEXTBOX #ref, X, Y [, width, height, FC, BC]
GUI BCOLOUR colour, #ref1 [, #ref2, ...]
GUI BEEP msec
GUI DELETE #ref1 [, #ref2, ...] | ALL
GUI DISABLE #ref1 [, #ref2, ...] | ALL
GUI ENABLE #ref1 [, #ref2, ...] | ALL
GUI FCOLOUR colour, #ref1 [, #ref2, ...]
GUI HIDE #ref1 [, #ref2, ...] | ALL
GUI NUMBERBOX CANCEL
GUI REDRAW #ref1 [, #ref2, ...] | ALL
GUI SHOW #ref1 [, #ref2, ...] | ALL
GUI TEXTBOX CANCEL
GUI INTERRUPT down [, up]
ctrl = TOUCH(DOWN | UP | LASTX | LASTY | REF | LASTREF)
val = CTRLVAL(#ref) CTRLVAL(#ref) = value
GUI SETUP #n
PAGE #n [, #n2, ...]
button = MSGBOX (msg\$, b1\$, b2\$ [, b3\$ [, b4\$]])

Commands

' (single quotation mark) - comment
? (question mark) - shorthand for PRINT
CLEAR
CONST id1 = expression [, id2 = expression, ...]
CONTINUE DO | FOR
DATA constant[, constant]...
DATE\$ = "DD-MM-YY" | "DD/MM/YY"
DIM [type] var [, var, ...] [AS type [, var AS type, ...]]
DO [WHILE <test>]
LOOP
DO
LOOP UNTIL <test>
ERASE array [, array, ...]
ERROR [message\$]
EXIT DO | FOR | FUNCTION | SUB
FOR var = start TO finish [STEP increment]
NEXT [var1 [, var2, ...]
FUNCTION name (arg1 [, arg2, ...]) [AS <type>]
END FUNCTION
GOSUB target
RETURN
GOTO target
IF <test> THEN <stmt> ELSE <stmts>
IF <test> THEN - ELSEIF - ELSE - ENDIF
INPUT ["prompt string\$";] [,] var [, var, ...]
LINE INPUT ["prompt string\$";] var\$
LET variable = expression
variable = expression
LOCAL [type] decl [, decl, ...] [AS type [, var AS type, ...]]
ON ERROR ABORT | IGNORE | SKIP [nn] | CLEAR
ON nbr GOTO | GOSUB target1 [, target2, ...]
ON KEY subroutine
PAUSE ms
PRINT expression1 [, ;] [expression2, ...] [, ;]
RANDOMIZE nbr
READ var1 [, var2, ...]
RESTORE [line]
REM comment
SELECT CASE - CASE [ELSE] - END SELECT
SETTICK period, target [, nbr]
SUB name arg1 [, arg2, ...]
END SUB
TIME\$ = "HH:MM:SS" | "HH:MM" | "HH"

Communications & File I/O

OPEN C\$ AS #nbr
C\$ = "COMn: baud, buf, int, nbr, DE, 9BIT, INV, OC, S2"
I2C OPEN speed, timeout [, PU]
I2C WRITE addr, option, sendlen, data [, data ...]
I2C READ addr, option, rcvlen, rcvbuf
I2C SLAVE OPEN addr, mask, opt, i_send, i_rcv
I2C SLAVE WRITE len, data [, data ...]
I2C SLAVE READ len, buf, rcvd
I2C [SLAVE] CLOSE
ONEWIRE READ pin, flag, len, data, ...
ONEWIRE WRITE pin, flag, len, data, ...
ONEWIRE RESET pin
SPI[2] OPEN speed, mode, bits
received_data = SPI[2](data_to_send)
SPI[2] WRITE nbr, data1[, ...] | str\$ | array()
SPI[2] READ nbr, array()
SPI[2] CLOSE
OPTION SD CARD CS [, CD [, WP]] | DISABLE
OPEN fname\$ FOR mode AS #fibr
'mode' = INPUT | OUTPUT | APPEND | RANDOM
LOAD file\$ [, R]
MKDIR dir\$
CHDIR dir\$
NAME old\$ AS new\$
SAVE [file\$]
SEEK [#]fibr, pos
fname\$ = DIR\$([fspec [, type]])
CLOSE [#]fibr [, [#]fibr] ...
State = EOF([#]fibr)
INPUT #fibr, var1 [, var2, ...]
LINE INPUT #fibr, string variable\$
PRINT #fibr, expression1 [, ;] [expression2, ...] [, ;]
INPUT\$(nbr, [#]fibr)
nbr = LOC([#]fibr) nbr = LOF([#]fibr)
PLAY TONE left [, right [, duration]]
PLAY WAV file\$ [, interrupt]
PLAY PAUSE | RESUME | STOP | VOLUME left, right

Micromite MMBasic V5.4

(Micromite Plus features are in red)

Downloads: <http://geoffg.net/micromite.html>

Forum: <http://www.thebackshed.com/forum/Microcontrollers>

Copyright Geoff Graham, 2017

Distributed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Australia license (CC BY-NC-SA 3.0)

Functions

ACOS(nbr)	ABS(nbr)
ASIN(nbr)	ATN(nbr)
COS(nbr)	DEG(radians)
EXP(nbr)	LOG(nbr)
PI	RAD(degrees)
SIN(nbr)	SQR(nbr)
TAN(nbr)	EVAL(str\$)
CINT(nbr)	FIX(nbr)
INT(nbr)	
ASC(str\$)	BIN\$(nbr [, chars])
CHR\$(nbr)	FIELD\$(str\$, field, delim\$)
HEX\$(nbr [, chars])	INSTR(start, str\$, pat\$)
LEFT\$(str\$, nbr)	RIGHT\$(str\$, nbr)
LEN(str\$)	MID\$(str\$, start [, nbr])
OCT\$(nbr [, chars])	SPACE\$(nbr)
STR\$(nbr [, m [, n [, c\$]])	
STRING\$(nbr, ascii [, str\$)	
LCASE\$(str\$)	UCASE\$(str\$)
VAL(str\$)	
DATE\$	TIME\$
TIMER	INKEY\$
MAX(nbr [, nbr [, ...]])	MIN(nbr [, nbr [, ...]])
POS	RND(nbr)
SGN(nbr)	TAB(nbr)

Options

OPTION AUTORUN OFF | ON
OPTION BASE 0 | 1
OPTION BAUDRATE nbr
OPTION BREAK nn
OPTION CASE UPPER | LOWER | TITLE
OPTION CLOACKTRIM ±n
OPTION COLOURCODE ON | OFF
OPTION CONSOLE ECHO | NOECHO
OPTION CONSOLE INVERT | NOINVERT
OPTION CONSOLE AUTO
OPTION DEFAULT FLOAT | INTEGER | STRING | NONE
OPTION DISPLAY lines [, chars]
OPTION ERROR CONTINUE | ABORT
OPTION EXPLICIT
OPTION KEYBOARD nn
OPTION LIST
OPTION PIN nbr
OPTION RESET
OPTION TAB 2 | 4 | 8

Devices

IR dev, key, int | CLOSE
KEYPAD var, int, r1, r2, r3, r4, c1, c2, c3, c4 | CLOSE
LCD INIT d4, d5, d6, d7, rs, en
LCD line, pos, text\$ | CLEAR | CLOSE
LCD CMD | DATA d1 [, d2 [, etc]]
PWM channel, freq, pwm1 [, pwm2 [, pwm3]]
PWM channel, STOP
RTC GETTIME
RTC SETTIME year, month, day, hour, minute, second
RTC SETREG | GETREG register, value | var
OPTION RTC data, clock | DISABLE
SERVO channel [, freq], out1 [, out2 [, out3]]
SERVO channel, STOP
TEMPR START pin [, precision 0 to 3]
Temperature = TEMPR(pin)

LCD Display Panel

OPTION LCDPANEL ctrl, orient, D/C, reset [, CS]
ctrl = ILI9163 | ST7735 | ILI9341
OPTION LCDPANEL ctrl, orient [, LCD-A] [, readpin]
ctrl = SSD1963 [, 4]] [5]] [7]] [7A]] [8]
OPTION LCDPANEL CONSOLE [font [, fc [, bc [, blight]]]]
OPTION LCDPANEL NOCONSOLE
OPTION LCDPANEL DISABLE
GUI CALIBRATE
GUI RESET LCDPANEL
GUI TEST LCDPANEL | TEST TOUCH
OPTION TOUCH_T_CS pin, T_IRQ pin [, click pin]
OPTION TOUCH DISABLE
PIXEL x, y [, colour]
LINE x1, y1, x2, y2 [, lw [, colour]]
CIRCLE x, y, r [, lw] [, a] [, colour] [, fill]
TRIANGLE x1, y1, x2, y2, x3, y3 [, colour [, FILL]]
BOX x1, y1, w, h [, lw] [, colour] [, fill]
RBOX x1, y1, w, h [, rc] [, colour] [, fill]
TEXT x, y, str\$ [, just\$] [, fnt] [, scale] [, colour] [, bc]
GUI BITMAP x, y, data [, w] [, h] [, s] [, colour] [, bc]
CLS [colour]
COLOUR fore [, back]
COLOR fore [, back]
FONT [#]font-number, scaling
BACKLIGHT percent
BLIT READ | WRITE [#]buffer, x, y, w, h
BLIT CLOSE [#]buffer
BLIT x1, y1, x2, y2, w, h
colour% = RGB(red, green, blue | colour listed below)
white black blue green cyan red magenta yellow brown gray
coordinate = TOUCH(X | Y)