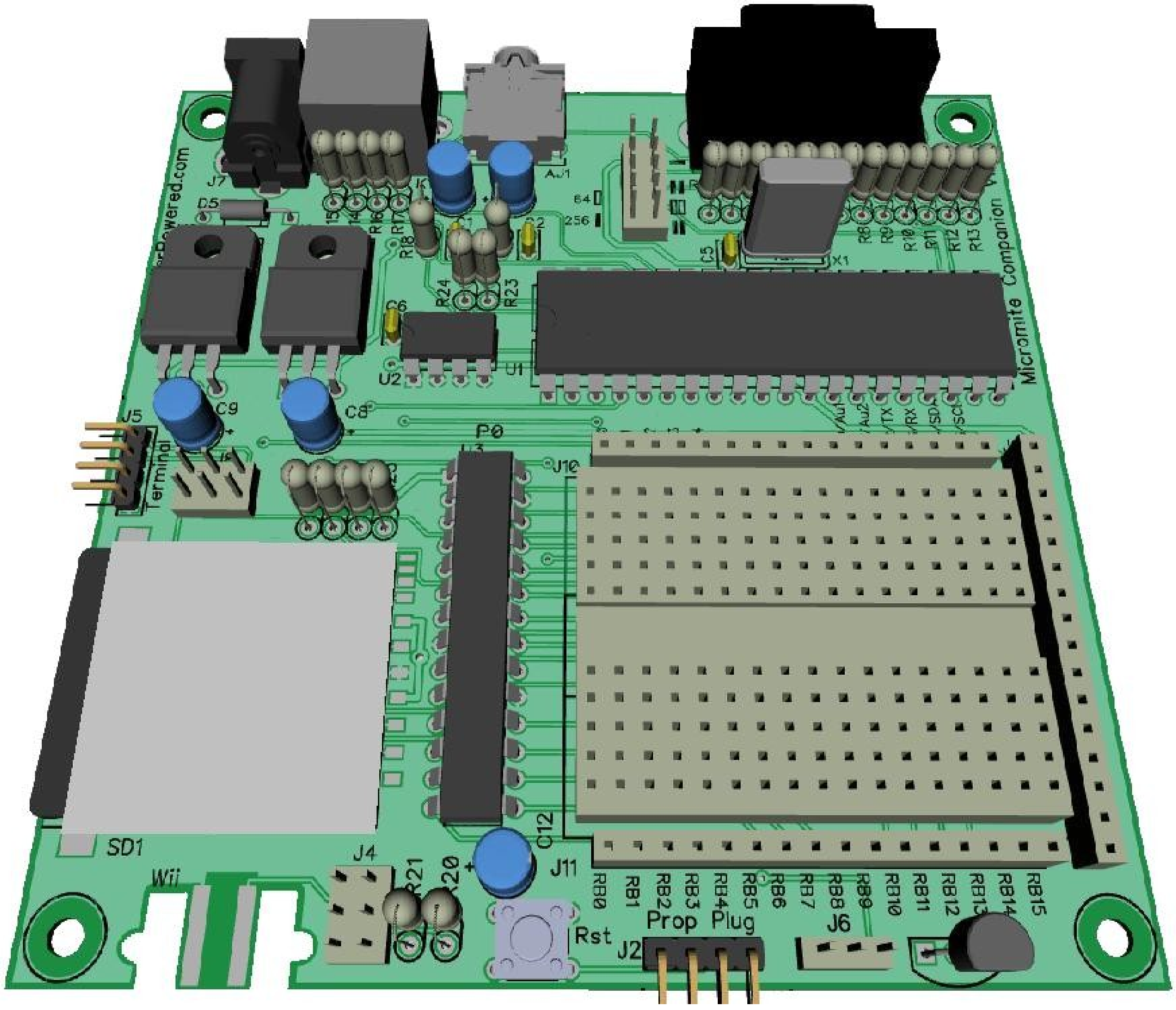


Micromite Companion Kit

Concept and Software by: Jeff Ledger
Board designed by: Karl Albrecht

http://www.propellerpowered.com

Product Support: http://forums.propellerpowered.com

CREDITS AND DISCLAIMER:

The “Micromite” PIC chip code was created by Geoffrey R Graham.
Propeller chip was created by Chip Gracey, distributed by Parallax, Inc.
MMBASIC was created by Geoffrey R Graham.
GWBASIC & QBASIC are copyright © Microsoft Corporation

The Micromite Companion's Propeller firmware would not have been possible without Open Source object contributions from the following individuals:

Chip Gracey
Chris Gadd
Johannes Ahlebrand
Juergen Buchmueller
Marco Maccaferri
Marko Lukat
Tomas Rokicki

The Micromite Companion Kit is designed for entertainment purposes only! It was not designed for “mission critical” operations or industrial/commercial use. While the chances of injury from using this kit are nearly non-existent, The builder of the kit takes full responsibility for his safety while both assembling the kit, and using it. **The designers of the hardware and software at not responsible for damages resulting from the use or misuse of all materials involved.**

Introduction:

Design inspired by the 8bit computers of yesterday, the Micromite Companion Kit is a powerful, single-board microcomputer powered by two unique microcontrollers. The VGA video and audio outputs, PS/2 keyboard interface, and SD socket are controlled and maintained by the Parallax Propeller. At the helm of this powerful input/output system is the Micromite, a PIC chip programmed with a modern BASIC interpreter (MMBASIC) with built-in editing software.   The “fuel-oxidizer” combination of these two microcontrollers creates an explosive combination that is capable of both retro-style entertainment or futuristic functionality in single kit.

The **Micromite Companion Kit** is perfect for the following individuals:

*** The “retro” enthusiast looking for a great project to enjoy or share early computing.**

The powerful MMBASIC interpreter is capable of running BASIC programs written over the last three decades, many times with little to no modification. MMBASIC accepts GWBASIC style “line numbered” code, as well as QBASIC style “numberless” code.   In addition, the Propeller is BASIC accessible as an I2C video/audio device allowing for 16/64/256 color modes, sprites, and tiles. The Micromite MMBASIC is capable of executing up to 23,000 lines of code per second.

*** The “beginner” who is interested to learning microcontrollers.**

The Micromite's MMBASIC provides simple BASIC commands to control modern microcontroller projects including LEDs, servos, sensors, LCD displays using modern connection protocols which include serial, One-wire, I2C, and SPI.

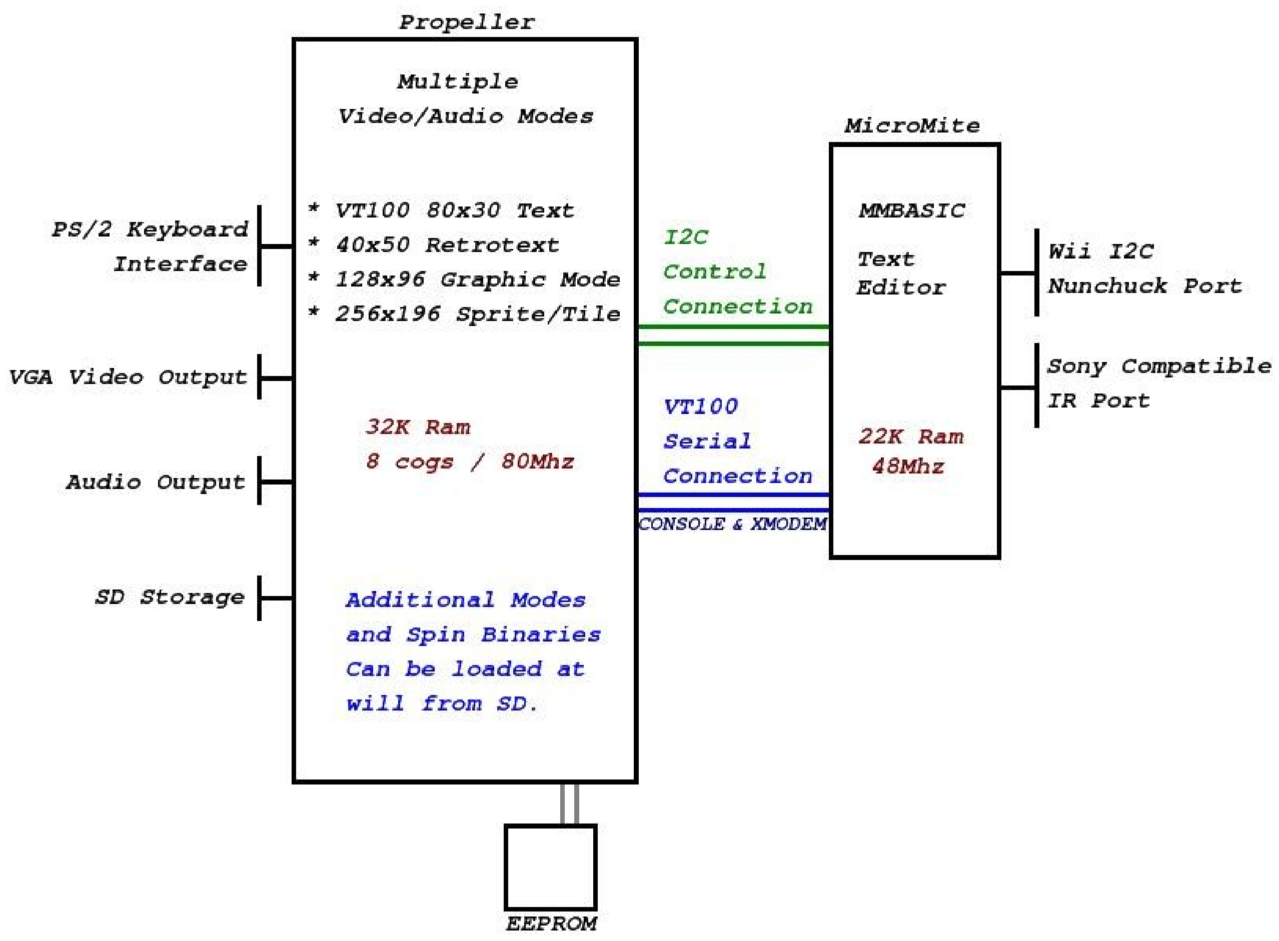
*** The “hardware hobbyist” who is looking for an easy way to control hardware projects.**

TTL Serial programming connections for both the Parallax Propeller and the Micromite are provided, allowing the Micromite Companion Kit to go in completely new directions.   A breadboard experimentation area contains easy connections to all of the I/O pins of the Micromite as well as the open I/O of the Parallax Propeller.

*** The “expert” who is looking to expand their knowledge into other microcontrollers.**

The Propeller, directly programmable in it's native languages of Spin and Assembly, can also be programmed in C due to the recent release of a C compiler. The kit's interfaces to VGA, audio, keyboard, and SD providing an excellent range of ready devices for learning.

Micromite Companion Microcontroller Overview



At power-up, the Propeller, acting as primary input/output device loads it's initial program from the 32K storage EEPROM. This firmware is a sophisticated VT100 terminal program which connects to the console pins of the Micromite, simultaneously opening an I2C “command” channel as well..

At the same time, the Micromite starts a 38400 baud serial connection on it's console pins and sends a “>” prompt which the Propeller passes forward to the VGA display.

The Propeller may be commanded to change video/audio modes at will by either the user pressing CTRL-F1 – CTRL-F8 on the keyboard, or by accepting special I2C command controls over the I2C connection.

Switching modes by either key-press or by I2C command will require that the “*mode#.mde*” binary file be present on the SD card. At time of release, the Micromite Companion firmware is designed with 4 different graphics/audio modes. The ability to load a new mode at will from the SD card, gives the kit potentially unlimited capability.

At any time, pressing the reset button will halt the Propeller, causing the Propeller to re-load it's initial 80x30, VT100 terminal/text mode from it's EEPROM. **It's possible to still have a Micromite program running when switching back to Companion mode. CTRL-C will stop the BASIC program. If you don't see a > prompt, it's probably means a BASIC program still running.**

Hardware Requirements:

The Micromite Companion Kit requires the following items:

* A PS/2 compatible 101 or 104 keyboard. *USB Keyboards with PS/2 adapters should also work.*

* A VGA monitor capable of 1024x768 resolution. *Most modern monitors can go well above this.*

* A "center positive" 7.5v-9v, 1amp, DC power supply with 2.1mm barrel connection

* A 500MB – 2GB Secure Digital Card or MicroSD card with adapter. *2GB max*

* A set of amplified PC speakers. *For ear safety, headphones are not recommended.*

Optional: *(but recommended for full enjoyment)*

* A Wii Nunchuck Controller

* A remote control capable of Sony IR codes

* Electronics (LEDs, LCD display, servos, RTC chip, etc)

* A Parallax "Propplug"

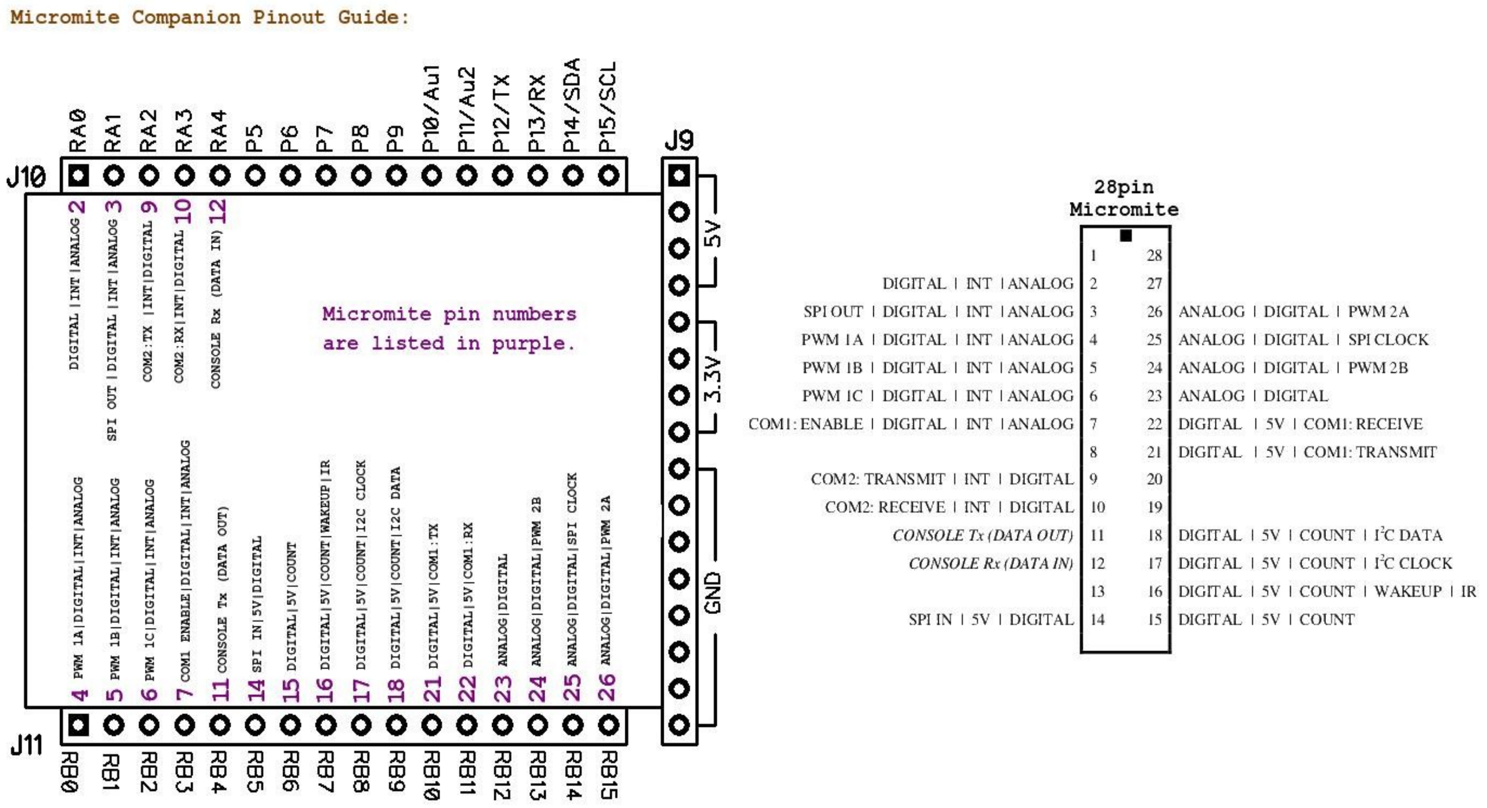
Special Note! ***Before reading further...***

It is advisable to start with the "Micromite Manual" written by Geoff Graham before continuing further in this material. The Micromite Manual will familiarize you with the both the hardware features of the Micromite chip itself and the MMBASIC programming language.

The Micromite Companion Kit comes pre-loaded with the Propeller Micromite Companion firmware required to behave as a VT100 terminal, allowing you to use the on-board editor and MMBASIC.

Once you have familiarized yourself with the MMBASIC and the Micromite chip itself, this manual will provide you both the hardware map to the Micromite breadboard section *(next page)*, the guide to the Secure Digital (SD) file system commands, and the advanced I2C controls for access to the Propeller's graphics and audio capabilities.

Micromite Companion Pinout Guide:


Micromite Companion Pinout Guide:
J10
RA0
RA1
RA2
RA3
RA4
P5
P6
P7
P8
P9
P10/Au1
P11/Au2
P12/TX
P13/RX
P14/SDA
P15/SCL
2 DIGITAL | INT | ANALOG
3 SPI OUT | DIGITAL | INT | ANALOG
9 COM2:TX | INT | DIGITAL
10 COM2:RX | INT | DIGITAL
12 CONSOLE Rx (DATA IN)
Micromite pin numbers are listed in purple.
4 PWM 1A | DIGITAL | INT | ANALOG
5 PWM 1B | DIGITAL | INT | ANALOG
6 PWM 1C | DIGITAL | INT | ANALOG
7 COM1 ENABLE | DIGITAL | INT | ANALOG
11 CONSOLE Tx (DATA OUT)
14 SPI IN | 5V | DIGITAL
15 DIGITAL | 5V | COUNT
16 DIGITAL | 5V | COUNT | WAKEUP | IR
17 DIGITAL | 5V | COUNT | I2C CLOCK
18 DIGITAL | 5V | COUNT | I2C DATA
21 DIGITAL | 5V | COM1:TX
22 DIGITAL | 5V | COM1:RX
23 ANALOG | DIGITAL
24 ANALOG | DIGITAL | PWM 2B
25 ANALOG | DIGITAL | SPI CLOCK
26 ANALOG | DIGITAL | PWM 2A
J11
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RB8
RB9
RB10
RB11
RB12
RB13
RB14
RB15
J9
5V
3.3V
GND
28pin Micromite
1
28
DIGITAL | INT | ANALOG 2
27
SPI OUT | DIGITAL | INT | ANALOG 3
26 ANALOG | DIGITAL | PWM 2A
PWM 1A | DIGITAL | INT | ANALOG 4
25 ANALOG | DIGITAL | SPI CLOCK
PWM 1B | DIGITAL | INT | ANALOG 5
24 ANALOG | DIGITAL | PWM 2B
PWM 1C | DIGITAL | INT | ANALOG 6
23 ANALOG | DIGITAL
COM1: ENABLE | DIGITAL | INT | ANALOG 7
22 DIGITAL | 5V | COM1: RECEIVE
8
21 DIGITAL | 5V | COM1: TRANSMIT
COM2: TRANSMIT | INT | DIGITAL 9
20
COM2: RECEIVE | INT | DIGITAL 10
19
CONSOLE Tx (DATA OUT) 11
18 DIGITAL | 5V | COUNT | I²C DATA
CONSOLE Rx (DATA IN) 12
17 DIGITAL | 5V | COUNT | I²C CLOCK
13
16 DIGITAL | 5V | COUNT | WAKEUP | IR
SPI IN | 5V | DIGITAL 14
15 DIGITAL | 5V | COUNT

The Micromite Companion provides breadboard access to all of the Micromite's I/O pins.
The guide above will help you translate the 28pin Micromite pin numbers used in the Micromite documentation to the RB and RA pin-out used by the Microchip PIC32 datasheet.

Access to the unused Propeller I/O is provided as P5 – P9. In addition you can tap into P10 – P15. *(Depending which mode you are in, P10-15 may or may not be used by the Propeller.)*

The Wii Nunchuck and IR ports are connected to the Micromite's I2C & IR connections internally.

Warning! The Propeller & Micromite are 3.3v devices. Take care to never put 5v on a Propeller I/O or Micromite 3v pin without a resistor to limit the voltage. Seven pins on the Micromite are tolerant to 5v and are listed as such.

MODE 1 – Companion Mode:

When the Micromite Companion is powered on by inserting plugging in it's power supply, it starts in "Companion Mode." Companion Mode is an 80x30, 8 color, VT100 compatible "terminal" mode that is specifically designed to provide access to the on-board editor of the Micromite chip, as well as provide access to the SDOS (SD Operating System) commands.

<u>SDOS Commands available in Companion Mode:</u>

DIR (or CTRL-D)

DIR shows the file contents on the root of your SD card. There are several special types of files which can be identified by both presented color as well as their file extension. Files use the 8.3 format.

.BAS = MMBASIC Programs	.BIN = Executable Binary Programs
.FNT = Mode4 font/tiles	.SPR = Mode4 sprites
.SCR = Mode4 screen layout	.MDE = *{not displayed with DIR}* Mode Files

LOAD (or CTRL-L)

LOAD preforms a built-in XMODEM transfer between the Propeller and the Micromite to transfer programs from the SD card into the memory of the Micromite. Generally, .BAS files are loaded, but the ASCII .FNT, .SCR, and .SCR files may also be loaded for editing. Files use the 8.3 format.

Example:

```
LOAD<ENTER>TEST.BAS<ENTER>
CTRL-L TEST.BAS<ENTER>
```

SAVE (or CTRL-S)

SAVE preforms a built-in XMODEM transfer between the Propeller and the Micromite to transfer a program from the memory of the Micromite to the SD card. Generally, .BAS files are saved, but an ASCII file may also be saved. Files use the 8.3 format.

Example:

```
SAVE<ENTER>TEST.BAS<ENTER>
CTRL-S TEST.BAS<ENTER>
```

BRUN (or CTRL-B)

BRUN is used to load and run binary (.BIN) files. Binary files may the extension .BIN or .MDE, but if not extension is provided, the .BIN extension is assumed. Binary files are compiled Propeller binary files which were created by Propeller Tool or BST using the Propeller's C or Spin languages.

Example:

```
BRUN<ENTER>TEST.BIN<ENTER>
BRUN<ENTER>TEST<ENTER>
CTRL-B TEST.BIN<ENTER>
```

F11 {Erase File}

The F11 key will prompt to “Enter filename to delete.” Typing the complete 'filename.extension' will delete the file from your SD card. Files use the 8.3 format.

Example: F11 TEST.BAS<ENTER>

Other Micromite Function Key Shortcuts can be called from Companion Mode:

F2	=	RUN program in Micromite memory.
F3	=	LIST program in Micromite memory.
F4	=	EDIT program in Micromite memory.

Note: EDIT has additional F-key commands. Refer to the Micromite Manual for more information.

Shortcut Function Keys:

CTRL-F2 – CTRL-F8 are mode shortcut keys. These are used to BRUN binary files mode{#}.bin if they are present on your SD card. In the assembly instructions your were told to copy three mode{#}.bin files to your SD card. (mode2.bin, mode3.bin, mode4.bin) Mode{#}.bin files can be launched using CTRL-F{#} keys or using I2C mode commands from inside your MMBASIC programs.

Possible usage is as follows:

CTRL-F2	=	mode2.bin
CTRL-F3	=	mode3.bin
CTRL-F4	=	mode4.bin
CTRL-F5	=	mode5.bin
CTRL-F6	=	mode6.bin
CTRL-F7	=	mode7.bin
CTRL-F8	=	mode8.bin

Early versions of Micromite Companion used these mode exclusively to switch to other graphics modes. We have added .MDE files in place of these required .BIN files, allowing you to assign mode{X}.bin file to any Propeller binary you desire. Initially, it's not a bad idea to have the original *(mode2.bin, mode3.bin, and mode4.bin)* files on your SD card just to maintain compatibility with some of the older program examples.

.MDE files are officially designated and distributed by Propellerpowered to maintain compatibility for all Micromite Companion users. Up to 254 .MDE modes are possible.

MODE 2 – RETROTEXT MODE

SD Requirements: mode2.mde from Propellerpowered.com
mode2.bin from Propellerpowered.com (for older programming examples)

Mode2 is an Atari/Commodore styled 40x50, 64 color text mode which resembles computer which were popular in the 1980's. It contains many special I2C commands which provide access to enhanced graphics and sound abilities. All of the existing MMBASIC commands are also available in mode2.

Once you've issued a command from MMBASIC to switch to Mode2, console commands are directly compatible. (with the exception of EDIT which must be used in Companion Mode).

Example program:

```
I2C OPEN 400,100                  ' Open the I2C channel on the Micromite
I2C WRITE &h42,0,3,1,2,2          ' Switch to mode2.mde.  {last digit indicates mode2}
PAUSE 2000                        ' Give the Propeller time to switch modes.
I2C WRITE &h42,0,4,1,205,63,0     ' Change text color to 63 {white} and background to 0 {black}
PAUSE 10                          ' A pause is a good idea when using I2C commands in mode2.

FOR X = 1 TO 20
  PRINT “HELLO WORLD”
NEXT X
```

You should familiar yourself with the I2C section of the Micromite Manual before using the special I2C commands which allow you to control the Propeller's extended video and audio features.

Two I2C Examples for Mode2:

I2C WRITE &h42,0,3,1,2,2 *{this command is issued while in Companion Mode}*

&h42	=	I2C address of the Propeller chip
3	=	Number of commands given. (1,2,2 = 3 commands)
1	=	Tell the Propeller an I2C command is about to be given.
2	=	Tell the Propeller a .mde mode switch is about to happen.
2	=	Tell the Propeller which mode to switch to.

I2C WRITE &h42,0,4,1,205,63,0

&h42	=	I2C address of the Propeller chip
4	=	Number of commands given. (1,205,63,0 = 4 commands)
1	=	Tell the Propeller an I2C command is about to be given.
205	=	Tell the Propeller to change text & background colors.
63	=	Text color
0	=	Background color

MODE 2 – RETROTEXT MODE

Note: I2C controls 180-189 control reading & writing to/from the SD card.
See FSRW.BAS and WRITE.BAS for examples of these commands in use.
Commands 180-197 are also available from Companion Mode, mode1.

180 Read filename from data on I2C registers 2-13
I2C WRITE &h42,0,2,1,180

It's generally a good idea to clear the I2C registers first (230) before using this command.
A for next loop is the easiest way to move a filename string to the I2C registers. (see FSRW.BAS)

181 Command Propeller to open SD file for reading. (Requires 180 to be used first)
I2C WRITE &h42,0,2,1,181

182 Command Propeller to open SD file for writing. (Requires 180 to be used first)
I2C WRITE &h42,0,2,1,182

183 Command Propeller to open SD file for append. (Requires 180 to be used first)
I2C WRITE &h42,0,2,1,183

185 Command Propeller to write byte from I2C register 1 to SD file.
I2C WRITE &h42,0,2,1,185

Requires that a byte was written to I2C register 1 before using this command.

186 Command Propeller to read byte from SD card and place it on I2C register 1.
I2C WRITE &h42,0,2,1,186

Register 1 can be read with the following example right after command 186.

```
I2C WRITE &h42,0,1,&h0
I2C READ &h42,0,1,a$
```

187 Command Propeller to boot at binary program from the SD card. (requires 180)
I2C WRITE &h42,0,2,1,187

189 Command Propeller to close the opened file on the SD card when reading or writing.
I2C WRITE &h42,0,2,1,189

MODE 2 – RETROTEXT MODE

190 Reboot the Propeller, re-launching Companion Mode.
I2C WRITE &h42,0,2,1,190

This does not stop a running program in the Micromite chip.
If a > prompt is not presented, hit CTRL-C to halt the running BASIC program if desired.

191 Command Propeller to switch to mode2.bin
I2C WRITE &h42,0,2,1,191 *Always use a pause (PAUSE 2000) after using this command.*

192 Command Propeller to switch to mode3.bin
I2C WRITE &h42,0,2,1,192 *Always use a pause (PAUSE 2000) after using this command.*

193 Command Propeller to switch to mode4.bin
I2C WRITE &h42,0,2,1,193 *Always use a pause (PAUSE 2000) after using this command.*

194 Command Propeller to switch to mode5.bin
I2C WRITE &h42,0,2,1,194 *Always use a pause (PAUSE 2000) after using this command.*

195 Command Propeller to switch to mode6.bin
I2C WRITE &h42,0,2,1,195 *Always use a pause (PAUSE 2000) after using this command.*

196 Command Propeller to switch to mode7.bin
I2C WRITE &h42,0,2,1,196 *Always use a pause (PAUSE 2000) after using this command.*

197 Command Propeller to switch to mode8.bin
I2C WRITE &h42,0,2,1,197 *Always use a pause (PAUSE 2000) after using this command.*

200 Character Redefine (Valid on ASCII characters 1-127)
I2C WRITE &h42,0,11,1,200,character,row1,row2,row3,row4,row5,row6,row7,row8

Example: Changes ! To a smiley face.
I2C WRITE &h42,0,11,1,200,33,60,66,165,129,165,189,66,60
PAUSE 1000 'A 1 second pause is required after issuing this command.

201 Plot a block at an X and Y position
I2C WRITE &h42,0,5,1,201,X,Y,COLOR

202 Plot a Line *A short pause after issuing this statement is a good idea.*
I2C WRITE &h42,0,7,1,X,Y,XX,YY,COLOR

204 Clear the Screen *A short pause after issuing this statement is a good idea.*
I2C WRITE &h42,0,2,1,204

205 Change Text & Background Colors *A short pause after issuing this statement is a good idea***.**
I2C WRITE &h42,0,4,1,205,textcolor,backgroundcolor

MODE 2 – RETROTEXT MODE

207 **Locate the cursor at a new X and Y.** *A short pause after this command is recommended.*
I2C WRITE &h42,4,1,207,X,Y

208 **Redefine all lowercase characters to PETSCII character set #1**
I2C WRITE &h42,0,2,1,208

209 **Redefine all lowercase characters to PETSCII character set #2**
I2C WRITE &h42,0,2,1,209

220 **Turn on Tiny Synth Audio**
I2C WRITE &h42,0,2,1,220

221 **Play a Tiny Synth audio note**
I2C WRITE &h42,0,3,1,221,note

222 **Turn off Tiny Synth Audio**
I2C WRITE &h42,0,2,1,222

223 **Change Tiny Synth Decay, Sustain, Pulse Width, PWM rate, and Fill.**
I2C WRITE &h42,0,7,1,223,decay,sustain,pulsewidth,pwm,fill

230 **Flush all I2C registers**
I2C WRITE &h42,0,2,1,230

MODE 3 – SLUG GRAPHICS MODE

SD Requirements: mode3.mde from Propellerpowered.com
mode3.bin from Propellerpowered.com (for older programming examples)

Mode3 is a 64 color block pixel mode which resembles game systems which were popular in the 1970's. It contains many special I2C commands which provide access to enhanced graphics and sound abilities. All of the existing MMBASIC commands are also available in mode3.

Once you've issued a command from MMBASIC to switch to Mode3, console commands are directly compatible. (with the exception of EDIT which must be used in Companion Mode).

Example program:

```
I2C OPEN 400,100              ' Open the I2C channel on the Micromite
I2C WRITE &h42,0,3,1,2,3      ' Switch to mode3.mde.  {last digit indicates mode3}
PAUSE 2000                    ' Give the Propeller time to switch modes.
I2C WRITE &h42,0,4,1,205,63,0 ' Change pixel color to 63 {white} and background to 0 {black}
PAUSE 10                      ' A pause is a good idea when using I2C commands in mode3.

FOR X = 1 TO 25
  I2C WRITE &h42,0,5,1,201,x,x,x  'Draw a multicolor line using X=X, Y=X, and COLOR=X
  PAUSE 10                        ' A pause is a good idea when using I2C commands in mode3.
NEXT X
```

You should familiar yourself with the I2C section of the Micromite Manual before using the special I2C commands which allow you to control the Propeller's extended video and audio features.

<u>Two I2C Examples for Mode3:</u>

I2C WRITE &h42,0,3,1,2,3 *{this command is issued while in Companion Mode}*

&h42	=	I2C address of the Propeller chip
3	=	Number of commands given. (1,2,3 = 3 commands)
1	=	Tell the Propeller an I2C command is about to be given.
2	=	Tell the Propeller a .mde mode switch is about to happen.
3	=	Tell the Propeller which mode to switch to.

I2C WRITE &h42,0,4,1,205,63,0

&h42	=	I2C address of the Propeller chip
4	=	Number of commands given. (1,205,63,0 = 4 commands)
1	=	Tell the Propeller an I2C command is about to be given.
205	=	Tell the Propeller to change text & background colors.
63	=	Pixel color
0	=	Background color

MODE 3 – SLUG GRAPHICS MODE

Note: I2C controls 180-189 control reading & writing to/from the SD card.
See FSRW.BAS and WRITE.BAS for examples of these commands in use.

180 Read filename from data on I2C registers 2-13
I2C WRITE &h42,0,2,1,180

It's generally a good idea to clear the I2C registers first (230) before using this command.
A for next loop is the easiest way to move a filename string to the I2C registers. (see FSRW.BAS)

181 Command Propeller to open SD file for reading. (Requires 180 to be used first)
I2C WRITE &h42,0,2,1,181

182 Command Propeller to open SD file for writing. (Requires 180 to be used first)
I2C WRITE &h42,0,2,1,182

183 Command Propeller to open SD file for append. (Requires 180 to be used first)
I2C WRITE &h42,0,2,1,183

185 Command Propeller to write byte from I2C register 1 to SD file.
I2C WRITE &h42,0,2,1,185

Requires that a byte was written to I2C register 1 before using this command.

186 Command Propeller to read byte from SD card and place it on I2C register 1.
I2C WRITE &h42,0,2,1,186

Register 1 can be read with the following example right after command 186.

```
I2C WRITE &h42,0,1,&h0
I2C READ &h42,0,1,a$
```

187 Command Propeller to boot at binary program from the SD card. (requires 180)
I2C WRITE &h42,0,2,1,187

189 Command Propeller to close the opened file on the SD card when reading or writing.
I2C WRITE &h42,0,2,1,189

MODE 3 – SLUG GRAPHICS MODE

190 Reboot the Propeller, re-launching Companion Mode.
I2C WRITE &h42,0,2,1,190

This does not stop a running program in the Micromite chip.
If a > prompt is not presented, hit CTRL-C to halt the running BASIC program if desired.

191 Command Propeller to switch to mode2.bin
I2C WRITE &h42,0,2,1,191 *Always use a pause (PAUSE 2000) after using this command.*

192 Command Propeller to switch to mode3.bin
I2C WRITE &h42,0,2,1,192 *Always use a pause (PAUSE 2000) after using this command.*

193 Command Propeller to switch to mode4.bin
I2C WRITE &h42,0,2,1,193 *Always use a pause (PAUSE 2000) after using this command.*

194 Command Propeller to switch to mode5.bin
I2C WRITE &h42,0,2,1,194 *Always use a pause (PAUSE 2000) after using this command.*

195 Command Propeller to switch to mode6.bin
I2C WRITE &h42,0,2,1,195 *Always use a pause (PAUSE 2000) after using this command.*

196 Command Propeller to switch to mode7.bin
I2C WRITE &h42,0,2,1,196 *Always use a pause (PAUSE 2000) after using this command.*

197 Command Propeller to switch to mode8.bin
I2C WRITE &h42,0,2,1,197 *Always use a pause (PAUSE 2000) after using this command.*

200 Character Redefine (Valid on ASCII characters 1-127)
I2C WRITE &h42,0,11,1,200,character,row1,row2,row3,row4,row5,row6,row7,row8

Example: Changes ! To a smiley face.
I2C WRITE &h42,0,11,1,200,33,60,66,165,129,165,189,66,60
PAUSE 1000 'A 1 second pause is required after issuing this command.

201 Plot a block at an X and Y position
I2C WRITE &h42,0,5,1,201,X,Y,COLOR

202 Plot a Line *A short pause after issuing this statement is a good idea.*
I2C WRITE &h42,0,7,1,X,Y,XX,YY,COLOR

204 Clear the Screen *A short pause after issuing this statement is a good idea.*
I2C WRITE &h42,0,2,1,204

205 Change Text & Background Colors *A short pause after issuing this statement is a good idea*.
I2C WRITE &h42,0,4,1,205,textcolor,backgroundcolor

MODE 3 – SLUG GRAPHICS MODE

220 **Turn on Tiny Synth Audio**
I2C WRITE &h42,0,2,1,220

221 **Play a Tiny Synth audio note**
I2C WRITE &h42,0,3,1,221,note

222 **Turn off Tiny Synth Audio**
I2C WRITE &h42,0,2,1,222

223 **Change Tiny Synth Decay, Sustain, Pulse Width, PWM rate, and Fill.**
I2C WRITE &h42,0,7,1,223,decay,sustain,pulsewidth,pwm,fill

230 **Flush all I2C registers**
I2C WRITE &h42,0,2,1,230

MODE 4 – PGE SPRITE/TILE GRAPHICS

SD Requirements: mode4.mde from Propellerpowered.com
mode4.bin from Propellerpowered.com (for older programming examples)

Mode4 is an 256x192, 16x16 sprite, 8x8 tile, 64 color sprite/tile mode which resembles arcade machines which were popular in the 80's and 90's. It contains many special I2C commands which provide access to enhanced graphics abilities. All of the existing MMBASIC commands are also available in mode4.

Once you've issued a command from MMBASIC to switch to Mode4, the only console commands which are valid are CTRL-C (stop the program) and F4 switch back to Companion Mode, EDIT.

Mode4 can be both the most challenging, yet the most rewarding mode to work with.
It is advisable to study the BASIC Examples in the software library on Propellerpowered.com

This section provide you both the basic I2C control commands along with recommended Micromite subroutines which will make Mode4 programs much easier to create and debug.

Example program:

```
I2C OPEN 400,100                              ' Open the I2C channel on the Micromite
I2C WRITE &h42,0,3,1,2,4                      ' Switch to mode4.mde. {last digit indicates mode4}
DO WHILE ASC(A$)<>42 : A$=inkey$: LOOP        ' Wait for the Propeller to switch modes.

FOR X = 1 TO 20
  I2C WRITE PC,0,6,1,200,10,X,X,X             ' Displays character tiles 1-20 if already loaded.
NEXT X
```

You should familiar yourself with the I2C section of the Micromite Manual before using the special I2C commands which allow you to control the Propeller's extended video features.

You should also practice using I2C controls using mode2 before attempting mode4.

<u>Two I2C Examples for Mode2:</u>

I2C WRITE &h42,0,3,1,2,4 *{this command is issued while in Companion Mode}*

&h42	=	I2C address of the Propeller chip
3	=	Number of commands given. (1,2,4 = 3 commands)
1	=	Tell the Propeller an I2C command is about to be given.
2	=	Tell the Propeller a .mde mode switch is about to happen.
4	=	Tell the Propeller which mode to switch to.

MODE 4 – PGE SPRITE/TILE GRAPHICS

180 Read filename from data on I2C registers 2-13 and load Sprite Resources from SD.
I2C WRITE &h42,0,2,1,180

181 Read filename from data on I2C registers 2-13 and load Tile Resources from SD.
I2C WRITE &h42,0,2,1,181

182 Read filename from data on I2C registers 2-13 and load Screen Resources from SD.
I2C WRITE &h42,0,2,1,182

Commands 180, 181, and 182 may be invoked multiple time during the course of a Mode4 program, allowing the programmer the ability to change the sprites, tiles, or background screen at any time to expand or enhance the program.

It is strongly advisable to use a subroutine *(provided at the bottom of page)* to use 180,181,and 182.

180 Sprite Resource File – May be edited/created with EDIT or SPREDIT.BIN

This ASCII file contains sixteen 16x16 sprites which may be up to 16 colors.
The ASCII .SPR file sprite data is # (hash) followed by 16 hexadecimal numbers (colors) 0-F

```
' Example Sprite
#0000000000000000
#0000000000F00000
#FFFFF0000FFFF000
#0FFFFF000FF8EEE0
#00FFFFF00FFFEE00
#000FFFFF0FFF0000
#0000FFFFFF000000
#00FFFFFFFF000000
#0FFFFFFFFF000000
#000FFFFF00000000
#000000EE00000000
#0000EE0000000000
#0000000000000000
#0000000000000000
#0000000000000000
#0000000000000000
```

Recommended Micromite Subroutine for loading a sprite resource file.

```
LoadResource "demo.spr",180

SUB LoadResource file$,cmd
  I2C WRITE &h42,0,2,1,230 'Clear the I2C registers
  FOR X = 1 TO LEN(file$) 'Send filename
     I2C WRITE PC,0,2,x+1,ASC(MID$(file$,x,1)) : NEXT X
  I2C WRITE &h42,0,2,1,cmd 'Initate Load 180, 181, or 182
  a$ ="": DO WHILE a$ <> "*" : a$=INKEY$ : LOOP : PAUSE 1000
END SUB
```

MODE 4 – PGE SPRITE/TILE GRAPHICS

181 Character/Tile Resource File – May be edited/created with EDIT

This ASCII file contains sixtytwo 8x8 character tiles which may be up to 16 colors.
The ASCII .FNT file tile data is # (hash) followed by 16 hexadecimal numbers (colors) 0-F

```
'' TILE 11 = A
#000FFF00
#00FF0FF0
#0FF000FF
#0FF000FF
#0FFFFFFF
#0FF000FF
#0FF000FF
#00000000
```

Tile 0 corresponds to space, or blank and shouldn't be changed.
(This doesn't mean that you can't. It just may lead to unexpected results.)

Tiles 1-10 correspond to the numbers 0-9

Tiles 11-35 correspond to UPPERCASE A-Z

Tile 37 corresponds to – (minus sign)

Tiles 38-62 correspond to lowercase a-z

You are free to change the tiles to anything required providing you adhere to the #00000000 format. Generally, I've found it handy to maintain tiles 0-37 to allow for display of instructional text or scoreboard on the screen, then define tiles 38-62 (a-z) any way that I please.

Recommended Micromite Subroutine for loading a character/tile resource file.

```
LoadResource "demo.fnt",181

SUB LoadResource file$,cmd
  I2C WRITE &h42,0,2,1,230 'Clear the I2C registers
  FOR X = 1 TO LEN(file$) 'Send filename
     I2C WRITE PC,0,2,x+1,ASC(MID$(file$,x,1)) : NEXT X
  I2C WRITE &h42,0,2,1,cmd 'Initate Load 180, 181, or 182
  a$ ="": DO WHILE a$ <> "*" : a$=INKEY$ : LOOP : PAUSE 1000
END SUB
```

MODE 4 – PGE SPRITE/TILE GRAPHICS

182 Background Screen Resource File – May be edited/created with EDIT

This ASCII file 31 lines of 28 characters, each line proceeded by a # (hash)
The ASCII **.SCR** file tile data is # (hash) followed by ascii numbers or letters (a-z) or (A-Z).

```
#abbbbbbbbbbbbbbbbbbbbbbbbbba
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#bbbbbbbbbbbbbbbbbbbbbbbbbbbb
#abcdefghijklmnopqrstuvwxyzba
```

The letters and numbers are referenced from the .FNT Character/Tile file. Each of the positions represent a visible character position on the screen. In this example, my lowercase "a" is a copyright symbol, so it is displayed in the upper/left and upper/right corners of the screen.

Recommended Micromite Subroutine for loading a character/tile resource file.

```
LoadResource "demo.scr",182

SUB LoadResource file$,cmd
  I2C WRITE &h42,0,2,1,230 'Clear the I2C registers
  FOR X = 1 TO LEN(file$) 'Send filename
     I2C WRITE PC,0,2,x+1,ASC(MID$(file$,x,1)) : NEXT X
  I2C WRITE &h42,0,2,1,cmd 'Initate Load 180, 181, or 182
  a$ ="": DO WHILE a$ <> "*" : a$=INKEY$ : LOOP : PAUSE 1000
END SUB
```

MODE 4 – PGE SPRITE/TILE GRAPHICS

190 Reboot the Propeller, re-launching Companion Mode.
I2C WRITE &h42,0,2,1,190

This does not stop a running program in the Micromite chip.
If a > prompt is not presented, hit CTRL-C to halt the running BASIC program if desired.

200 Write Byte at X,Y,Palette,Character
I2C write &h42,0,6,1,200,CharX,CharY,palette,char

Recommend Subroutines:

```
SUB DisplayChar CharX,CharY,palette,char
        I2C write &h42,0,6,1,200,CharX,CharY,palette,char
END SUB

SUB DisplayText textx,texty,palette,text$
         FOR x = 1 TO LEN(text$) : char = ASC(MID$(text$,x,1))
                   I2C write &h42,0,6,1,200,textx+x,texty,palette,char
          PAUSE 1 :NEXT x
END SUB
```

202 LoadSprite Sprite,Graphic,X,Y,mirror,palette
I2C WRITE &h42,0,8,1,202,sprite,graphic,x,y,mirror,palette

Recommended Subroutine:

```
SUB LoadSpr sprite,graphic,x,y,mirror,palette
        I2C WRITE &h42,0,8,1,202,sprite,graphic,x,y,mirror,palette
END SUB
```

203 Set Sprite Position Sprite,X,Y
I2C WRITE &h42,0,5,1,203,sprite,x,y

204 Hide Sprite #
I2C WRITE &h42,0,3,1,204,sprite

MODE 4 – PGE SPRITE/TILE GRAPHICS

219 Animate Sprite - Note: Sprite must be loaded with 202 first.)
I2C WRITE &h42,0,6,1,219,sprite,start_graphic,end_graphic,delay_time

Animate does not require pre-loading all the sprites in the sequence since they are already loaded in the Propeller's memory using 180. The sprite will animate using old-fashion page-flipping.

Recommended Subroutine:

```
SUB Animate sprite,start,end,delay
    I2C WRITE &h42,0,6,1,219,sprite,start,end,delay
    PAUSE 5
END SUB
```

220 MoveSpeed – Note: sprite must already be loaded with 202 first.)
I2C Write &h42,0,7,1,220,sprite,xdelay,ydelay,xinc,yinc

Movespeed will move either an animated or single sprite automatically on the screen. The sprite will continue to move in the direction you've assigned until you re-assign it again.

Recommended Subroutine:

```
SUB MoveSpeed sprite,xdelay,ydelay,xinc,yinc
    I2C Write &h42,0,7,1,220,sprite,xdelay,ydelay,xinc,yinc
END SUB
```