

Micromite eXtreme User Manual

MMBasic Ver 5.4.14

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

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

The Micromite eXtreme is a new addition to the Micromite family using the 64, 100 and 144-pin PIC32 MZ microcontrollers. The Micromite eXtreme firmware implements all the features of the standard Micromite and the Micromite Plus as described in the *Micromite User Manual* and the *Micromite Plus Manual*. It also has a number of additional features and they are described in this document.

The focus of this manual is to describe just the features that are **unique** to the Micromite eXtreme. For general Micromite programming you should refer to the *Micromite User Manual* and the *Micromite Plus Manual* in addition to this manual.

Contents

Introduction	4
Micromite Family Summary	5
Suitable Microcontrollers	7
Typical Circuit	8
Programming the Firmware	9
64-pin Micromite eXtreme Pinouts	10
100-pin Micromite eXtreme Pinouts	12
144-pin Micromite eXtreme Pinouts	15
VGA Driver	19
Mouse Support	20
Unique Micromite eXtreme Features	21
Commands (Micromite eXtreme Only)	23
Functions (Micromite eXtreme Only)	31
Appendix A - Sensor Fusion	34
Appendix B - Backpack144 GPIO pinout	35
Appendix C - Sprites	37

Introduction

This section provides an introduction for users who are familiar with the Micromite and the Micromite Plus and need a summary of the extra features in the Micromite eXtreme and Micromite eXtreme64.

The Micromite eXtreme is an extension of the standard Micromite and the Micromite Plus; all features of these two versions are also in the Micromite eXtreme. This includes features of the BASIC language, input/output, communications, etc. Some commands have changed slightly (for example the CPU command) but for the main part Micromite programs will run unchanged on the Micromite eXtreme.

The following summarises additional features in the Micromite eXtreme as compared to the standard Micromite and the Micromite Plus:

PIC32 MZ Processor

The Micromite eXtreme is based on the Microchip PIC32MZ 32 bit microcontroller. This chip is available in 64, 100 and 144-pin surface mount packages and is up to five times faster and has up to ten times the program space of the MX series used in the standard Micromite.

High Speed Double Precision Floating Point

The Micromite eXtreme uses the built in hardware floating point capability of the PIC32MZ which is much faster than floating point on the standard Micromite and uses double precision floating point.

I/O Pins

The 64-pin Micromite eXtreme has 46 free I/O pins with 24 analogue capable. The 100-pin Micromite eXtreme has up to 71 free I/O pins with 40 analog capable and the 144-pin chip has up to 115 free I/O pins with 50 analogue capable. All analogue pins use a 12-bit analogue to digital conversion rather than 10-bit on the standard Micromite.

The Micromite eXtreme has two I²C ports, three SPI ports, six PWM channels and up to four serial COM ports. All serial COM ports are high speed (over 1,000,000 baud).

The Micromite eXtreme64 has one I²C and two SPI ports but is otherwise the same.

High Speed LCD Panels

Like the Micromite Plus the Micromite eXtreme supports ten different sized LCD display panels from 1.44" to 8". In addition it can drive displays using the SSD1963 controller in 16-bit parallel mode to achieve an even greater display update speed. There is also a range of drivers for displays that use a memory based framebuffer. This allows complex updates to be written whilst minimising screen flashing and tearing.

VGA Output

The Micromite eXtreme can drive a VGA display in 640 x 480 pixels or 640 x 400 (widescreen) with eight colours. All the graphics commands and GUI controls available in the Micromite Plus will also work on the VGA output. This capability is not available on the Micromite eXtreme64.

Mouse Input

The Micromite eXtreme can support a PS2 mouse which can be used to activate on screen GUI controls. This feature will work with touch sensitive LCD displays (it works in parallel with the touch sensitivity) but it is especially useful with VGA monitors that do not normally incorporate a touch sensitive surface.

USB Keyboard Input

The Micromite eXtreme can support a USB keyboard. The software supports UK and US layouts (see OPTION USBKEYBOARD) and is automatically enabled when a keyboard is plugged in.

Sprites

The Micromite eXtreme supports a complete implementation of sprites including screen scrolling and collision detection

Micromite Family Summary

The Micromite Family consists of three major types, the standard Micromite, the Micromite Plus and the Micromite eXtreme. All use the same BASIC interpreter and have the same basic capabilities however they differ in the number of I/O pins, the amount of memory, the displays that they support and their intended use.

Standard Micromite	Comes in a 28-pin or 44-pin package and is designed for small embedded controller applications and supports small LCD display panels. The 28-pin version is particularly easy to use as it is easy to solder and can be plugged into a standard 28-pin IC socket.
Micromite Plus	This uses a 64-pin and 100-pin TQFP surface mount package and supports a wide range of touch sensitive LCD display panels from 1.44" to 8" in addition to the standard features of the Micromite. It is intended as a sophisticated controller with easy to create on-screen controls such as buttons, switches, etc.
Micromite eXtreme	This comes in 64, 100-pin and 144-pin TQFP surface mount packages. The eXtreme version has all the features of the other two Micromites but is faster and has a larger memory capacity plus the ability to drive a VGA monitor for a large screen display. It works as a powerful, self contained computer with its own BASIC interpreter and instant start-up.

	Micromite		Micromite Plus		Micromite eXtreme		
	28-pin DIP	44-pin SMD	64-pin SMD	100-pin SMD	100-pin SMD	144-pin SMD	64-pin SMD
Maximum CPU Speed	48 MHz	48 MHz	120 MHz	120 MHz	252MHz	252 MHz	252 MHz
Maximum BASIC Program Size	59 KB	59 KB	100 KB	100 KB	540 KB	540 KB	540 KB
RAM Memory Size	52 KB	52 KB	108 KB	108 KB	460 KB	460 KB	460 KB
Clock Speed (MHz)	5 to 48	5 to 48	5 to 120	5 to 120	200 to 252	200 to 252	200 to 252
Total Number of I/O pins	19	33	45	77	75	115	46
Number of Analog Inputs	10	13	28	28	40	48	24
Number of Serial I/O ports	2	2	3 or 4	3 or 4	3 or 4	3 or 4	3 or 4
Number of SPI Channels	1	1	2	2	3	3	2
Number of I ² C Channels	1	1	1 + RTC	1 + RTC	2 + RTC	2 + RTC	1 + RTC
Number of 1-Wire I/O pins	19	33	45	77	75	115	46
PWM or Servo Channels	5	5	5	5	6	6	6
Serial Console	✓	✓	✓	✓	✓	✓	✓
USB Console			✓	✓	✓	✓	✓
PS2 Keyboard and LCD Console			✓	✓	✓	✓	✓
SD Card Interface			✓	✓	✓	✓	✓
Supports ILI9341 LCD Displays	✓	✓	✓	✓	✓	✓	✓
Supports Ten LCD Panels from 1.44" to 8" (diameter)			✓	✓	✓	✓	✓
Supports VGA Displays					✓	✓	

Sound Output (WAV/tones)			✓	✓	✓	✓	✓
Supports PS2 Mouse Input					✓	✓	✓
Floating Point Precision	Single	Single	Single	Single	Double	Double	Double
Power Requirements	3.3V 30 mA	3.3V 30 mA	3.3V 80 mA	3.3V 80 mA	3.3V 160 mA	3.3V 160 mA	3.3V 160 mA

Suitable Microcontrollers

The microcontroller used in the Micromite eXtreme is the PIC32MZ EF series manufactured by Microchip. There are two chip sizes (100-pin and 144-pin) with two frequency specifications (200 MHz and 252 MHz).

The default clock speed of the Micromite eXtreme is 200 MHz however this can be changed with a configuration option to 252 MHz if required. The firmware will automatically adjust for either the 100 or 144 pin version.

The recommended chips are:

PIC32MZ2048EF064I/PT	64-pin TQFP package (0.5 mm pin pitch) – maximum speed 200 MHz
PIC32MZ2048EFH064-250I/PT	64-pin TQFP package (0.5 mm pin pitch) – maximum speed 252 MHz
PIC32MZ2048EFG100-I/PF	100-pin TQFP package (0.5 mm pin pitch) – maximum speed 200 MHz
PIC32MZ2048EFG144-I/PL	144-pin LQFP package (0.5 mm pin pitch) – maximum speed 200 MHz
PIC32MZ2048EFH100-250I/PF	100-pin TQFP package (0.5 mm pin pitch) – maximum speed 252 MHz
PIC32MZ2048EFH144-250I/PL	144-pin LQFP package (0.5 mm pin pitch) – maximum speed 252 MHz

The chips PIC32MZ2048EFM064-I/PT , PIC32MZ2048EFM100-I/PF and PIC32MZ2048EFM144-I/PL can also be used at 200 MHz. In addition the 0.4mm chips may be used but are harder to solder by hand (PT for 100-pin and PH for 144-pin)

See <http://microchip.com> for the data sheets.

64-pin Test and Development Board

This board is in development. See http://www.thebackshed.com/forum/forum_posts.asp?TID=9344&PN=1 for details and updates.

100-pin Test and Development Board

The best development board for the 100-pin 200 MHz chip is the SnadPIC MZ, PIC32MZ EF MCU Starter Kit. This can be ordered with either the PIC32MZ2048EFG100-I/PF or PIC32MZ2048EFH100-I/PF processor (both are similar, the latter has a CAN facility but that is not supported by MMBasic).

If you are developing your own board it would be worth using the SnadPIC board for guidance.

SnadPIC boards can be purchased from:

<http://www.microcontroller-board.com/snadpic-board-32-bit/24-snadpic-mz-pic32mz-ef-mcu-starter-kit-pic32mz2048efg100.html>

or

<http://www.ebay.com.au/itm/PIC32MZ-USB-OTG-Microchip-Development-Board-Starter-kit-SD-Card-SnadPIC-MZ-/181805050475>

or search ebay.com for "SnadPIC MZ"

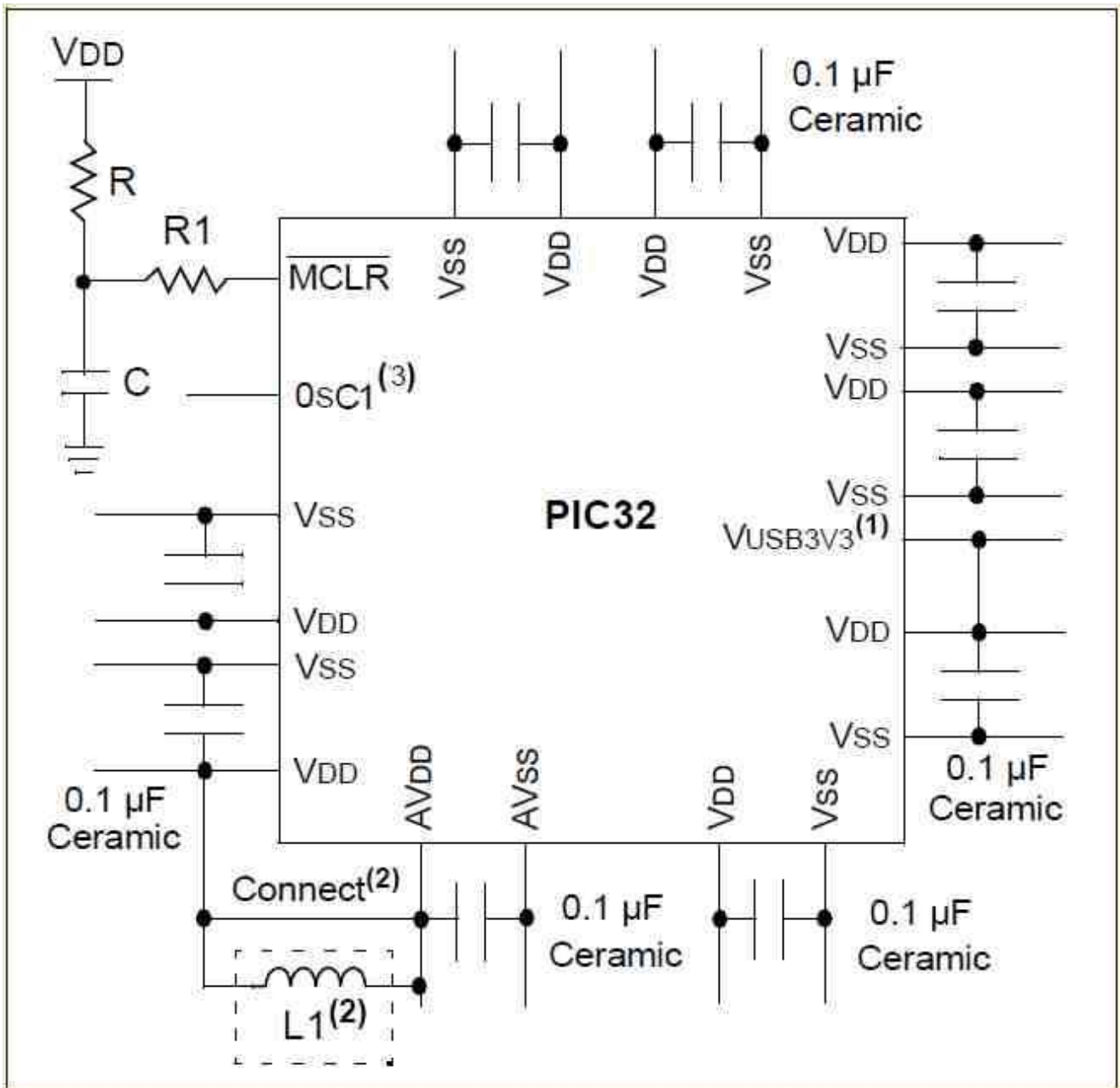
144-pin Test and Development Board

This board is a complete module and includes all the connectors for: VGA, TFT, PS2 keyboard, PS2 mouse, NunChuck, 3.5mm stereo sound, and USB. It has an onboard USB-to-UART (PIC16F1654), and sockets for an RTC module. It can be used standalone, or can be mounted directly onto the back of a 7" LCD display panel.

Available from <http://micromite.org>

Typical Circuit

An example of the required circuit for a Micromite eXtreme is given below:



Notes:

1. If the USB module is not used, this pin may be connected to VSS.
2. As an option, instead of a hard-wired connection, an inductor (L1) can be substituted between VDD and AVDD to improve ADC noise rejection. The inductor impedance should be less than 1ohm and the inductor capacity greater than 10 mA. Alternatively a 10ohm resistor could be substituted for L1. Typical values for R, R1, and C would be 10Kohm, 1Kohm, and 0.1uF
3. A 24MHz crystal oscillator must be connected to the OSC1 pin. e.g. Epson SG8002DCPHB24MHZ. See the pinout below for the pin number

Programming the Firmware

Programming the 64, 100 and 144-pin Micromite eXtreme is similar to programming the 28-pin standard Micromite described in the Micromite User Manual.

Refer to the following table for the pin connections to a PICKit 3 programmer:

PICKit 3 Pins	Description	64-pin Micromite eXtreme pin numbers	100-pin Micromite eXtreme pin numbers	144-pin Micromite eXtreme pin numbers
1 - MCLR	Master Reset (active low)	9	15	20
2 - Vcc	Power Supply (3.3V)	8, 26, 39, 54, 60, 19 (AVDD), 34 (VUSB3V3)	14, 37, 46, 62, 74, 83, 93, 30 (AVDD), 52 (VUSB3V3)	18, 33, 55, 64, 88, 107, 122, 137, 41 (AVDD), 74 (VUSB3V3)
3 - GND	Ground	7, 25, 35, 40, 55, 59	13, 36, 45, 53, 63, 75, 84, 92, 31 (AVSS)	17, 32, 54, 63, 75, 89, 108, 123, 136, 42(AVSS)
4 - PGD	Programming Data	16 or 18	25 or 27	36 or 38
5 - PGC	Programming Clock	15 or 17	24 or 26	35 or 37
6 - NC	Not used			

Notes:

- PDD/PGC must be used in matched pairs as aligned vertically in the table
- A pullup resistor of 10K is required between MCLR and Vcc.
- An oscillator is not required to program these chips and will be ignored if present
- The microcontroller being programmed can be powered by the PICKit 3 but it is recommended that a separate power supply be used. When the PICKit 3 supplies the power pin 2 (Vcc) on the PICKit 3 will become an output supplying power to the chip being programmed

64-pin Micromite eXtreme Pinouts

Pin	Features				
1	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D5	OV7670-D5
2	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D6	OV7670-D6
3	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D7	OV7670-D7
4	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SPI2-CLK	
5	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	I2C-SDA	
6	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	I2C-CLK	
7	VSS				
8	VDD				
9	MCLR				
10	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-1C	
11	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	OV7670-HR/RR
12	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	KBD-CLK	OV7670-VSYNC
13	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM2-RX	I2S-MCLK
14	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	
15	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	IR
16	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	OV7670-XC/WR
17	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM1-EN	OV7670-PC/RC
18	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-2B	
19	AVDD				
20	AVSS				
21	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-RESET	
22	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM1-RX	
23	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM2-TX	
24	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	KBD-DAT	
25	VSS				
26	VDD				
27	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-RS	
28	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-WR	
29	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM3-TX	I2S-BITCLK
30	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM3-RX	I2S-WORDCLK
31	OSC1				
32	DIGITAL_IN	DIGITAL_OUT	HEARTBEAT		
33	VBUS				
34	VDD				
35	VSS				
36	D-				
37	D+				

38	USBID				
39	VDD				
40	VSS				
41	DIGITAL_IN	DIGITAL_OUT	SPI2-OUT		
42	DIGITAL_IN	DIGITAL_OUT	COM1-TX		
43	DIGITAL_IN	DIGITAL_OUT	MOUSE-CLK		
44	DIGITAL_IN	DIGITAL_OUT	PWM-2A		
45	DIGITAL_IN	DIGITAL_OUT	PWM-2C	SOUND-LEFT	I2S-DATA
46	DIGITAL_IN	DIGITAL_OUT	PWM-1B		
47	DIGITAL_IN	DIGITAL_OUT	SPI2-IN		
48	DIGITAL_IN	DIGITAL_OUT	PWM-1A		
49	DIGITAL_IN	DIGITAL_OUT	SPI-CLK		
50	DIGITAL_IN	DIGITAL_OUT	SPI-IN		
51	DIGITAL_IN	DIGITAL_OUT	SPI-OUT		
52	DIGITAL_IN	DIGITAL_OUT	MOUSE-DAT		
53	DIGITAL_IN	DIGITAL_OUT	SOUND-RIGHT		
54	VDD				
55	VSS				
56	DIGITAL_IN	DIGITAL_OUT	COM4-TX	CONSOLE-TX	
57	DIGITAL_IN	DIGITAL_OUT	COM4-RX	CONSOLE-RX	
58	DIGITAL_IN	DIGITAL_OUT	SSD1963-D0	OV7670-D0	
59	VSS				
60	VDD				
61	DIGITAL_IN	DIGITAL_OUT	SSD1963-D1	OV7670-D1	
62	DIGITAL_IN	DIGITAL_OUT	SSD1963-D2	OV7670-D2	
63	DIGITAL_IN	DIGITAL_OUT	SSD1963-D3	OV7670-D3	
64	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D4	OV7670-D4

100-pin Micromite eXtreme Pinouts

Pin	Features				
1	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
2	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
3	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-2C	SOUND-LEFT
4	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
5	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
6	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	PWM-2A
7	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	
8	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	
9	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT	IR
10	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SPI2-CLK	
11	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	I2C-SDA	
12	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	I2C-CLK	
13	VSS				
14	VDD				
15	MCLR				
16	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-1C	
17	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
18	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SOUND-RIGHT	
19	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-BLU-SS	
20	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D5	
21	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D4	VGA-VSYNC
22	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D3	
23	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D2	
24	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D1	
25	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D0	
26	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D6	
27	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D7	
28	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
29	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
30	AVDD				
31	AVSS				
32	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D8	VGA-GRN-OUT
33	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D9	VGA-BLU-OUT
34	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D10	VGA-RED-OUT
35	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D11	
36	VSS				
37	VDD				

38	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT		
39	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM1-EN	VGA-BLU-CLK
40	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-2B	
41	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D12	
42	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D13	
43	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D14	VGA-RED-CLK
44	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-D15	VGA-RED-SS
45	VSS				
46	VDD				
47	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM1-RX	
48	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-GRN-CLK	
49	OSC1				
50	OSC2 - unused				
51	VBUS				
52	VDD				
53	VSS				
54	D-				
55	D+				
56	USBID				
57	DIGITAL_IN	DIGITAL_OUT	COM3-TX		
58	DIGITAL_IN	DIGITAL_OUT	COM3-RX		
59	DIGITAL_IN	DIGITAL_OUT	Snadpic-SD-CD	I2C2-CLK	
60	DIGITAL_IN	DIGITAL_OUT	I2C2-SDA		
61	DIGITAL_IN	DIGITAL_OUT			
62	VDD				
63	VSS				
64	DIGITAL_IN	DIGITAL_OUT	VGA-GRN-SS		
65	DIGITAL_IN	DIGITAL_OUT	COM1-TX		
66	DIGITAL_IN	DIGITAL_OUT	SPI2-OUT		
67	DIGITAL_IN	DIGITAL_OUT	SPI3-OUT	I2S-DATA	
68	DIGITAL_IN	DIGITAL_OUT	I2S-WORDCLK		
69	DIGITAL_IN	DIGITAL_OUT	SPI3-CLK	I2S-BITCLK	
70	DIGITAL_IN	DIGITAL_OUT	SPI3-IN	I2S-MCLK	
71	DIGITAL_IN	DIGITAL_OUT	PWM-1B		
72	DIGITAL_IN	DIGITAL_OUT	SPI2-IN		
73	DIGITAL_IN	DIGITAL_OUT	PWM-1A		
74	VDD				
75	VSS				
76	DIGITAL_IN	DIGITAL_OUT	SPI-CLK		
77	DIGITAL_IN	DIGITAL_OUT	SPI-IN		

78	DIGITAL_IN	DIGITAL_OUT	SPI-OUT	
79	DIGITAL_IN	DIGITAL_OUT	VGA-HSYNC	
80	DIGITAL_IN	DIGITAL_OUT		
81	DIGITAL_IN	DIGITAL_OUT	Snadpic-SD-CS	
82	DIGITAL_IN	DIGITAL_OUT		
83	VDD			
84	VSS			
85	DIGITAL_IN	DIGITAL_OUT	COM4-TX	CONSOLE-TX
86	DIGITAL_IN	DIGITAL_OUT	COM4-RX	CONSOLE-RX
87	DIGITAL_IN	DIGITAL_OUT	COM2-TX	
88	DIGITAL_IN	DIGITAL_OUT	COM2-RX	
89	DIGITAL_IN	DIGITAL_OUT	KBD-CLK	
90	DIGITAL_IN	DIGITAL_OUT	KBD-DAT	
91	DIGITAL_IN	DIGITAL_OUT	MOUSE-CLK	
92	VSS			
93	VDD			
94	DIGITAL_IN	DIGITAL_OUT	MOUSE-DAT	
95	DIGITAL_IN	DIGITAL_OUT	SSD1963-RESET	
96	DIGITAL_IN	DIGITAL_OUT	SSD1963-RS	
97	DIGITAL_IN	DIGITAL_OUT	SSD1963-WR	
98	DIGITAL_IN	DIGITAL_OUT		
99	DIGITAL_IN	DIGITAL_OUT	HEARTBEAT	
100	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	

144-pin Micromite eXtreme Pinouts

Pin	Features			
1	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
2	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
3	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-2C
4	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
5	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
6	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT3
7	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB8
8	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB9
9	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB12	
10	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB10	
11	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT1
12	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT2
13	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COUNT4-IR
14	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SPI2 CLK
15	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	I2C-SDA
16	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	I2C-CLK
17	GND			
18	VCC			
19	DIGITAL_IN	DIGITAL_OUT	HEARTBEAT	
20	RESET			
21	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
22	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
23	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM1-EN
24	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-HSYNC
25	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
26	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-VSYNC
27	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB11
28	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB13	
29	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB14	
30	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB15	
31	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
32	GND			
33	VCC			
34	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SOUND-LEFT
35	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	SOUND-RIGHT
36	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	CAMERA-XCLK
37	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	

38	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
39	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
40	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
41	AVDD			
42	AVSS			
43	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	CAMERA-D0
44	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	CAMERA-D1
45	DIGITAL_IN	DIGITAL_OUT	CAMERA-D2	
46	DIGITAL_IN	DIGITAL_OUT	CAMERA-D3	
47	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-GRN-OUT
48	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-BLU-OUT
49	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-RED-OUT
50	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
51	DIGITAL_IN	DIGITAL_OUT		
52	DIGITAL_IN	DIGITAL_OUT		
53	DIGITAL_IN	DIGITAL_OUT		
54	GND			
55	VCC			
56	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
57	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-CLK
58	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	PWM-2B
59	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
60	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	
61	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-CLK
62	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-HSYNC
63	GND			
64	VCC			
65	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	CAMERA-D4
66	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	CAMERA-D5
67	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	CAMERA-D6
68	DIGITAL_IN	DIGITAL_OUT	CAMERA-D7	
69	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	COM1-RX
70	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	VGA-CLK
71	OSC1			
72	OSC2 - unused			
73	VBUS			
74	VCC			
75	GND			
76	USBD-			
77	USBD+			

78	USBID			
79	DIGITAL_IN	DIGITAL_OUT	COM3-TX	
80	DIGITAL_IN	DIGITAL_OUT	COM3-RX	
81	DIGITAL_IN	DIGITAL_OUT	CAMERA-HSYNC	CAMERA-RCK
82	DIGITAL_IN	DIGITAL_OUT	CAMERA-VSYNC	
83	DIGITAL_IN	DIGITAL_OUT	CAMERA-PCLK	CAMERA-RRST
84	DIGITAL_IN	DIGITAL_OUT	CAMERA-WR	
85	DIGITAL_IN	DIGITAL_OUT	I2C2-SCK	
86	DIGITAL_IN	DIGITAL_OUT	I2C2-SDA	
87	DIGITAL_IN	DIGITAL_OUT		
88	VCC			
89	GND			
90	DIGITAL_IN	DIGITAL_OUT	VGA-HSYNC	
91	DIGITAL_IN	DIGITAL_OUT	COM1-TX	
92	DIGITAL_IN	DIGITAL_OUT		
93	DIGITAL_IN	DIGITAL_OUT		
94	DIGITAL_IN	DIGITAL_OUT		
95	DIGITAL_IN	DIGITAL_OUT	SPI2-OUT	
96	DIGITAL_IN	DIGITAL_OUT	SPI3-OUT	I2S-DATA
97	DIGITAL_IN	DIGITAL_OUT	I2S-WORDCLK	I2S-MCLK
98	DIGITAL_IN	DIGITAL_OUT	SPI3-CLK	I2S-BITCLK
99	DIGITAL_IN	DIGITAL_OUT	SPI3-IN	
100	DIGITAL_IN	DIGITAL_OUT		
101	DIGITAL_IN	DIGITAL_OUT		
102	DIGITAL_IN	DIGITAL_OUT		
103	DIGITAL_IN	DIGITAL_OUT		
104	DIGITAL_IN	DIGITAL_OUT	PWM-1B	
105	DIGITAL_IN	DIGITAL_OUT	SPI2-IN	
106	DIGITAL_IN	DIGITAL_OUT	PWM-1A	
107	VCC			
108	GND			
109	DIGITAL_IN	DIGITAL_OUT	SPI-CLK	
110	DIGITAL_IN	DIGITAL_OUT	SPI-IN	
111	DIGITAL_IN	DIGITAL_OUT	SPI-OUT	
112	DIGITAL_IN	DIGITAL_OUT	VGA-HSYNC	
113	DIGITAL_IN	DIGITAL_OUT		
114	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB0	
115	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB1	
116	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB2	
117	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB3	

118	DIGITAL_IN	DIGITAL_OUT		
119	DIGITAL_IN	DIGITAL_OUT	PWM-1C	
120	DIGITAL_IN	DIGITAL_OUT	PWM-2A	
121	DIGITAL_IN	DIGITAL_OUT		
122	VCC			
123	GND			
124	DIGITAL_IN	DIGITAL_OUT	Console-TX	COM4-TX
125	DIGITAL_IN	DIGITAL_OUT	Console-RX	COM4-RX
126	DIGITAL_IN	DIGITAL_OUT		
127	DIGITAL_IN	DIGITAL_OUT	COM2-TX	
128	DIGITAL_IN	DIGITAL_OUT	COM2-RX	
129	DIGITAL_IN	DIGITAL_OUT	KB-CLK	
130	DIGITAL_IN	DIGITAL_OUT	KB-DAT	
131	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB4	
132	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB5	
133	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB6	
134	DIGITAL_IN	DIGITAL_OUT	SSD1963-DB7	
135	DIGITAL_IN	DIGITAL_OUT	MOUSE-CLK	
136	GND			
137	VCC			
138	DIGITAL_IN	DIGITAL_OUT	MOUSE-DAT	
139	DIGITAL_IN	DIGITAL_OUT	SSD1963-RESET	
140	DIGITAL_IN	DIGITAL_OUT	SSD1963-RS	
141	DIGITAL_IN	DIGITAL_OUT	SSD1963-WR	
142	DIGITAL_IN	DIGITAL_OUT		
143	DIGITAL_IN	DIGITAL_OUT		
144	ANALOG_IN	DIGITAL_IN	DIGITAL_OUT	

VGA Driver

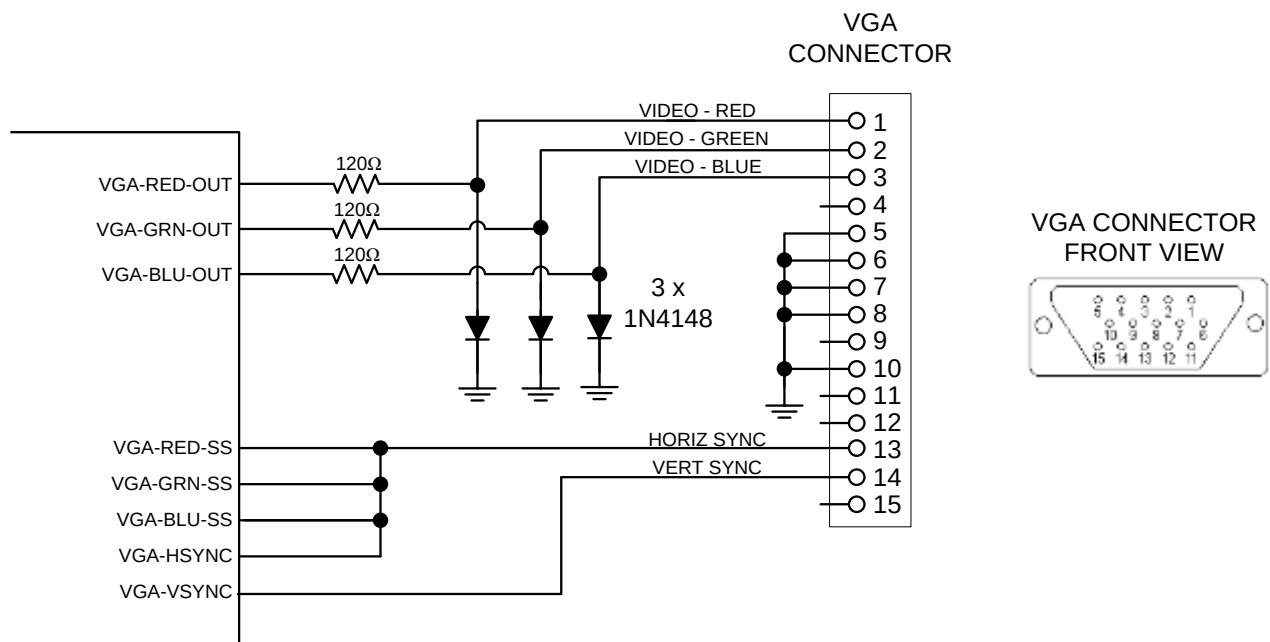
The Micromite eXtreme (but not the Micromite eXtreme64) can drive a standard VGA monitor by internally generating the necessary VGA signals (red, green, sync, etc). When a VGA monitor is connected and configured the VGA output works exactly the same as a connected LCD display panel – this means that all graphics commands, GUI objects, etc can be used as described in the Micromite and Micromite Plus manuals.

The features of the VGA driver are:

- 640 x 480 pixel output
- Optional 640 x 400 widescreen output
- Eight colours (red, blue, green, yellow, cyan, magenta, black, white)
- Works with all graphics and GUI commands.
- Supports any command/features that use transparency (transparent text and the BLIT command).
- Works as the console with 80 characters x 36 lines and will work with the EDIT command

VGA Connections

The connections for the VGA monitor are shown below:



The I/O pins VGA-RED-CLK, VGA-GRN-CLK, VGA-BLU-CLK must be left unconnected.

For the actual pin numbers refer to the pinout tables earlier in this manual.

Note that the diodes must be high speed signal types like the 1N4148 (not general purpose power diodes).

Configuring VGA Output

The command to enable the standard VGA 640 x 480 pixel output is:

```
OPTION LCDPANEL VGA
```

and to enable the widescreen format (640 x 400 pixels) the command is:

```
OPTION LCDPANEL VGA, 16
```

These commands only need to be run once as the parameters are stored in non volatile memory. Every time the Micromite is restarted MMBasic will automatically initialise the display ready for use. If the VGA output is no longer required the command `OPTION LCDPANEL DISABLE` can be used which will disable the VGA feature and return the I/O pins for general use.

If the VGA monitor is also to be used as the console the normal command can be used to enable this:

```
OPTION LCDPANEL CONSOLE
```

Mouse Support

The Micromite eXtreme supports a PS2 mouse which will act like a touch input on an LCD screen (it also works with the VGA output). MMBasic will automatically display a mouse pointer on the display which is moved by the mouse. When the left button the mouse is clicked it will act like a touch at the location pointed to by the cursor. This feature will also work with GUI controls.

Connecting the Mouse

The PS2 mouse uses a 6-pin DIN connector which should be connected to the Micromite eXtreme as illustrated.

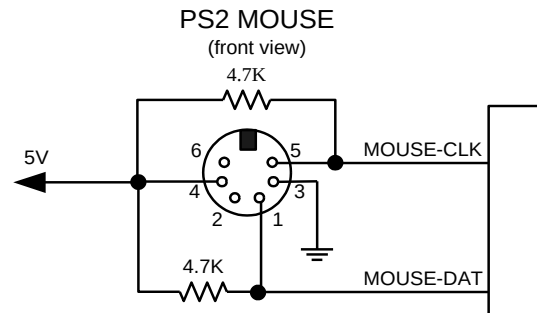
To enable the mouse the command is:

OPTION MOUSE ENABLE

and to disable it:

OPTION MOUSE DISABLE

These commands only need to be run once as the parameters are stored in non volatile memory. Every time the Micromite is restarted MMBasic will automatically initialise the mouse input ready for use.



Cursor Commands

The cursor (ie, mouse pointer) can be turned off/on, its colour can be set and its status can be overridden using the CURSOR command. It will only work with displays that support transparency. Displays capable of transparent text are a VGA monitor or any LCD panels that use the ILI9341 controller or an SSD1963 controller. The latter must have the RD pin specified in the OPTION LCDPANEL command. The cursor command can be used without a mouse to allow, for example, a joystick to move the cursor.

The cursor commands are as follows:

CURSOR ON

Enables the display of the cursor (this is the default when MOUSE is first enabled). If it is used after a previous CURSOR OFF command it will also restore the previous position of the cursor.

CURSOR OFF

Hides the cursor.

CURSOR X, Y [, LEFT] [, RIGHT] [, MID]

Positions the cursor to the screen location X and Y(in pixels) and sets the left-click (0 or 1) and optionally the right-click and mid-click status.

CURSOR COLOUR colour

Sets the cursor colour (this is a standard 24-bit colour value). CURSOR COLOR (US spelling) is also valid.

Cursor Functions

CURSOR(x)

Returns the current x coordinate (in pixels) of the cursor

CURSOR(y)

Returns the current y coordinate (in pixels) of the cursor

CURSOR(left)

Returns the current state of the left button

CURSOR(right)

Returns the current state of the right button

CURSOR(middle)

Returns the current state of the middle button

Unique Micromite eXtreme Features

Double Precision Floating Point

The Micromite eXtreme uses the hardware floating point capability of the MZ series of chips and can therefore process floating point calculations faster than the Micromite and Micromite Plus. All floating point uses double precision calculations.

Clock Speed Control

MMBasic can work with chips rated for 200 MHz or 252 MHz operation. By default the firmware will start running at 200 MHz however the 252 MHz clock speed can be selected with the command:

```
OPTION CPU 252
```

or the speed can be returned to 200 MHz with the command:

```
OPTION CPU 200
```

These commands change how MMBasic starts up and will cause a restart of the processor. The clock speed is saved in flash memory so the command only needs to be used once and will be automatically applied on startup.

Note: **OPTION CPU 252 must only be used on chips specifically rated for 250Mhz operation.** Use of this command on a 200MHz rated chip will cause MMBasic to stop running. Apart from the increased processing speed the only difference with a chip running at 252 MHz is the ability to use WAV files recorded at 24 KHz and 48 KHz.

16-bit Interface to SSD1963 Based LCD Displays

The Micromite eXtreme (but not the Micromite eXtreme64) can drive a SSD1963 display using a 16-bit parallel bus for extra speed. The extra I/O pins for this are listed as SSD1963-DB8 to SSD1963-DB15 on the pinout tables in this manual and they must be connected to the pins labelled DB8 to DB15 on the I/O connector on the SSD1963 display.

Note that in this mode the SSD1963 controller runs with a reduce colour range (65 thousand colours) compared to 16 million colours with the normal 8-bit interface.

To select the 16-bit bus the following controller names must be used with the OPTION LCDPANEL command when configuring the display:

- SSD1963_4_16 For a 4.3 inch display
- SSD1963_5_16 For a 5 inch display
- SSD1963_5A_16 For an alternative version of the 5 inch display if SSD1963_5 does not work
- SSD1963_7_16 For a 7 inch display
- SSD1963_7A_16 For an alternative version of the 7 inch display if SSD1963_7 does not work.
- SSD1963_8_16 For an 8 inch display.

Two I²C Channels

The Micromite eXtreme (but not the Micromite eXtreme64) supports two I²C channels. The second channel operates the same as the first, the only difference is that the commands use the notation I2C2 (for example I2C2 OPEN, etc).

Alternate SPI Channel for the SD Card

The SPI channel used for the SD Card (if configured) defaults to the second channel (SPI2) however this can be changed by appending the SPI channel number to the end of the OPTION SDCARD command this:

```
OPTION SDCARD CSPIN [,CDPIN] [,WPPIN] [,SPIno]
```

'SPIno' is the SPI controller to use and can be 1, 2 or 3.

This is particularly useful with the SnadPIC MZ, PIC32MZ EF MCU Starter Kit as it has the SD Card hardwired to controller 3. In that case the MMBasic command to configure the SD Card would be:

```
OPTION SDCARD 81, 59, , 3
```

Extended WAV File Playback

The Micromite eXtreme can play WAV files (like the Micromite Plus) however, if the eXtreme is configured to run at 252 Mhz it is also capable of playing WAV files recorded with sampling rates of 24 KHz and 48 KHz.

Six PWM Channels

The second PWM controller (ie, PWM 2) supports three channels (the other versions of the Micromite only support two). The command to use all three channels is:

```
PWM 2, freq, 2A, 2B, 2C
```

Similarly the SERVO command can also control six channels with the extra channel available on controller 2:

```
SERVO 2 [, freq], 2A, 2B, 2C
```

Three SPI Channels

The Micromite eXtreme (but not the Micromite eXtreme64) supports three SPI channels. The second and third channels operate the same as the first, the only difference is that the commands use the notation SPI2 and SPI3 (for example SPI3 WRITE, etc).

Note that by default, if the Micromite eXtreme is configured for a SPI based LCD panel, touch or an SD card then SPI2 will be unavailable to BASIC programs as these functions will use that channel.

Heartbeat

The heartbeat is an I/O pin which is pulsed off and on at a 1Hz rate. It is normally used to drive a LED to show that MMBasic is alive and running on the Micromite eXtreme.

The default is for it to be enabled however it can be disabled with:

```
OPTION HEARTBEAT DISABLE
```

If necessary it can be re enabled with:

```
OPTION HEARTBEAT ENABLE
```

These commands only needs to be run once as the parameters are stored in non volatile memory. Every time the Micromite is restarted MMBasic will automatically initialise the heartbeat feature.

Random Number Generation

The Micromite eXtreme uses the hardware random number generator in the MZ series of chips to deliver true random numbers. This means that the RANDOMIZE command is no longer needed and is not supported.

MM.DEVICE\$

On the Micromite eXtreme the read only variable MM.DEVICE\$ will return "Micromite eXtreme, Microchip ID 0xhhhhhhhh".

OPTION VCC command

The Micromite eXtreme supports the OPTION VCC command. This allows the user to precisely set the supply voltage to the chip and is used in the calculation of voltages when using analog inputs e.g. OPTION VCC 3.15. The parameter is not saved and should be initialised either on the command line or in a program.

CPU command

The Micromite eXtreme does not support dynamically changing the CPU speed or the sleep function.

Accordingly the commands CPU speed and CPU SLEEP are not available. However the eXtreme does support "CPU SLEEP time" where time is specified in seconds.

The CPU speed of the Micromite eXtreme can be permanently set to 200 MHz or 252 MHz using the OPTION CPU command.

OV7670 Camera Support

The Micromite eXtreme supports connection of an OV7670 camera

Longstring handling

The Micromite eXtreme supports a comprehensive set of commands and functions for handling longstrings stored in integer arrays

Commands (Micromite eXtreme Only)

BOX x1, y1, w, h [, lw] [,c] [,fill]	All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x1, y1, w, and h must all be arrays or all be single variables /constants otherwise an error will be generated. lw, c, and fill can be either arrays or single variables/constants. See the Micromite User manual for full details of parameter usage.
BLIT	The BLIT and SPRITE commands can be used interchangeably. All Micromite Plus functionality is preserved with minor functional changes. See the SPRITE command for functionality over and above the Micromite Plus and command differences.
CAMERA OPEN	Initialises an OV7670 camera ready for use
CAMERA OPEN FIFO	Initialises an OV7670 camera with FIFO ready for use
CAMERA CAPTURE	Captures an image from the camera to a connected 800x480 SSD1963 or 480x272 display. NB: the display must be set to 16-bit operation. Also available for the ILI9341 display on the 64-pin MMX
CAMERA SAVE "filename"	saves the on-screen image to the SDcard. If the file extension is not specified then ".BMP" is appended.
CAMERA REGISTER register, value	can be used to change the camera settings see the datasheet for details
CAMERA CLOSE	disables the camera and frees the allocated pins ILI9341 only AVAILABLE ON 64-PIN processor SSD1963 only AVAILABLE ON 144-PIN processor
CIRCLE x, y, r [,lw] [, a] [, c] [, fill]	All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x, y and r must all be arrays or all be single variables /constants otherwise an error will be generated. lw, a, c, and fill can be either arrays or single variables/constants. See the Micromite User manual for full details of parameter usage.
CLOSE [#]nbr [, [#]nbr]	The text "GPS" can be substituted for [#]nbr to close a communications port used for a GPS receiver
CURSOR ON	Enables the display of the cursor (this is the default when MOUSE is first enabled). If it is used after a previous CURSOR OFF command it will also restore the previous position of the cursor.
CURSOR OFF	Disables the display
CURSOR X, Y [, LEFT] [, RIGHT] [, MID]	Positions the cursor to the screen location X and Y(in pixels) and optionally sets the left-click (0 or 1), the right-click and mid-click status.
CURSOR COLOUR colour	Sets the cursor colour (this is a standard 24-bit colour value). CURSOR COLOR (US spelling) is also valid.

GUI STARTLINE n	Sets the row in the graphics memory which will appear at the top of the screen (landscape or reverse landscape) or left of the screen (portrait or reverse portrait) for a 4.3" SSD1963 display initialised with OPTION LCDPANEL SSD1963_4P [_16]
GUI VGA OFF GUI VGA ON	turns off the VGA output to allow a monitor to go to sleep turns the monitor back on again
I2C2 OPEN speed, timeout [, PU I2C2 WRITE addr, option, sendlen, senddata [,senddata] I2C2 READ addr, option, rcvlen, rcvbuf I2C2 CLOSE I2C2 SLAVE OPEN addr, mask, option, send_int, rcv_int I2C2 SLAVE WRITE sendlen, senddata [,senddata] I2C2 SLAVE READ rcvlen, rcvbuf, rcvd I2C2 SLAVE CLOSE	See Appendix B of the Micromite User Manual. Not available on the Micromite eXtreme64
LINE x1, y1, x2, y2 [, LW [, C]]	All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x1, y1, x2, and y2 must all be arrays or all be single variables /constants otherwise an error will be generated. lw and c can be either arrays or single variables/constants. See the Micromite User manual for full details of parameter usage.
LOAD SPRITE [#]n, fname\$	Alternate form. See SPRITE LOAD
LONGSTRING APPEND array%(), string\$ LONGSTRING CLEAR array%() LONGSTRING COPY dest%(), src%() LONGSTRING CONCAT dest%(), src%()	Append a normal MMBasic string to a long string variable. array%() is a long string variable while string\$ is a normal MMBasic string expression. Will clear the long string variable array%(). ie, it will be set to an empty string. Copy one long string to another. dest%() is the destination variable and src%() is the source variable. Whatever was in dest%() will be overwritten. Concatenate one long string to another. dest%() is the destination variable and src%() is the source variable. src%() will the added to the end of dest%() (the destination will not be

LONGSTRING LCASE array%()	overwritten). Will convert any uppercase characters in array%() to lowercase. array%() must be long string variable.
LONGSTRING LEFT dest%(), src%(), nbr	Will copy the left hand 'nbr' characters from src%() to dest%() overwriting whatever was in dest%(). ie, copy from the beginning of src%(). src%() and dest%() must be long string variables. 'nbr' must be an integer constant or expression.
LONGSTRING LOAD array%(), nbr, string\$	Will copy 'nbr' characters from string\$ to the long string variable array%() overwriting whatever was in array%().
LONGSTRING MID dest%(), src%(), start, nbr	Will copy 'nbr' characters from src%() to dest%() starting at character position 'start' overwriting whatever was in dest%(). ie, copy from the middle of src%(). 'nbr' is optional and if omitted the characters from 'start' to the end of the string will be copied src%() and dest%() must be long string variables. 'start' and 'nbr' must be an integer constants or expressions.
LONGSTRING REPLACE array%() , string\$, start	Will substitute characters in the normal MMBasic string string\$ into an existing long string array%() starting at position 'start' in the long string.
LONGSTRING RIGHT dest%(), src%(), nbr	Will copy the right hand 'nbr' characters from src%() to dest%() overwriting whatever was in dest%(). ie, copy from the end of src%(). src%() and dest%() must be long string variables. 'nbr' must be an integer constant or expression.
LONGSTRING TRIM array%(), nbr	Will trim 'nbr' characters from the left of a long string. array%() must be a long string variables. 'nbr' must be an integer constant or expression.
LONGSTRING UCASE array%()	Will convert any lowercase characters in array%() to uppercase. array%() must be long string variable.
OPEN comspec\$ AS GPS [,timezone_offset]	Will open a serial communications port for reading from a GPS receiver. See the GPS function for details. The timezone_offset parameter is used to convert UTC as received from the GPS to the local timezone. If omitted the timezone will default to UTC. The timezone_offset can be a any number between -12 and 14 allowing the time to be set correctly even for the Chatham Islands in New Zealand (UTC +12:45)
OPTION AUTOREFRESH mode	When using buffered driver TFT drivers this command controls when screen updates take place. When autorefresh is "ON" updates take place immediately. When autorefresh is "OFF" updates take place in the framebuffer and are only written to the screen when autorefresh is next turned "ON"

OPTION CPU speed	MMBasic can work with chips rated for 200 MHz or 252 MHz operation. By default the firmware will start running at 200 MHz however the 252 MHz clock speed can be selected with the command: OPTION CPU 252 or the speed can be returned to 200 MHz with the command: OPTION CPU 200 These commands change how MMBasic starts up and will cause a restart of the processor. The clock speed is saved in flash memory so the command only needs to be used once and will be automatically applied on startup. Note: OPTION CPU 252 must only be used on chips specifically rated for 250Mhz operation. Use of this command on a 200MHz rated chip will cause MMBasic to stop running. The chip will then need to be re-programmed with the Micromite eXtreme firmware. Apart from the increased processing speed the only difference with a chip running at 252 MHz is the ability to use WAV files recorded at 24 KHz and 48 KHz.
OPTION HEARTBEAT ENABLE OPTION HEARTBEAT DISABLE	Enables a heartbeat on an I/O pin which is pulsed on and off at a 1Hz rate. It is normally used to drive a LED to show that MMBasic is alive and running on the Micromite eXtreme. Disables the heartbeat
OPTION LCDPANEL ILI9341_16_BUFF, orientation [,RD pin]	Selects 16-bit bus operation of the ILI9341 display. Pin usage is exactly as per the SSD1963. Uses a memory resident framebuffer to improve performance. This uses 150K of memory leaving 319K for user programs.
OPTION LCDPANEL ILI9481_BUFF, orientation, DCpin, resetpin, chipselectpin	Selects SPI operation of a ILI9341 display using a memory resident framebuffer to improve performance. This uses 150K of memory leaving 319K for user programs.
OPTION LCDPANEL ILI9841, orientation, DCpin, RESETpin, CSpin	Initialises a TFT display using the ILI9841 controller. This supports 480 * 320 resolution. See the Micromite User manual, ILI9341 section, for full details of parameter usage.
OPTION LCDPANEL ILI9841_BUFF, orientation, DCpin, RESETpin, CSpin	Selects SPI operation of a ILI9481 display using a memory resident framebuffer to improve performance. This uses 300K of memory leaving 169K for user programs.
OPTION LCDPANEL SSD1963_n_16	Selects 16-bit bus operation of the various SSD1963 displays. See the Micromite plus manual for full details of syntax. Not available on the Micromite eXtreme64
OPTION LCDPANEL SSD1963_4_16_BUFF, orientation, [,RD pin]	Selects 16-bit bus operation of the 4.3" SSD1963 display using a memory framebuffer for improved performance. This uses 255K of memory leaving 213K for user programs Not available on the Micromite eXtreme64

OPTION LCDPANEL SSD1963_4P	Sets the 4.3" SSD1963 display up in 480 x 864 (landscape or reverse landscape) or 864x480 (portrait or reverse portrait) pixel mode. The screen viewport is 480x272 or 272x480 and the position of the viewport is controlled by GUI STARTLINE n. This mode of operation allows display updates to be done on a non-visible part of the graphics memory and then the viewport moved to see the updated image. The 4P display controller is fully compatible with TOUCH, MOUSE, CURSOR and GUI controls
OPTION LCDPANEL VGA [,16]	Enables output to a VGA display in 640 x 480 pixels or 640 x 400 (widescreen) with eight colours. All the graphics commands and GUI controls available in the Micromite Plus will also work on the VGA output. This command only needs to be run once as the parameters are stored in non volatile memory. When the Micromite is restarted the display will be automatically initialise ready for use. If the display is no longer required the command OPTION LCDPANEL DISABLE can be used to disable the VGA output and return the I/O pins for general use. Not available on the Micromite eXtreme64 .
OPTION MOUSE ENABLE	Enables mouse control of the cursor.
OPTION MOUSE DISABLE	Disables mouse control of the cursor and disables the cursor These commands only needs to be run once as the parameters are stored in non volatile memory. Every time the Micromite is restarted MMBasic will automatically initialise the mouse input ready for use.
OPTION SDCARD CSPIN [,CDPIN] [,WPPIN] [,SPIno]	The SPI channel used for the SD Card (if configured) defaults to the second channel (SPI2) however this can be changed by appending the SPI channel number to the end of the OPTION SDCARD
OPTION SERIAL PULLUP DISABLE	permanently stored option that disables pullups on all serial ports
OPTION SERIAL PULLUP ENABLE	default: permanently stored option that enables pullups on all serial ports
OPTION USBKEYBOARD layout	Sets the key layout for a connected USB keyboard. Valid layouts are UK and US
OPTION VCC voltage	This allows the user to precisely set the supply voltage to the chip and is used in the calculation of voltages when using analogue inputs. The parameter is not saved and should be initialised either on the command line or in a program.
PIXEL x, y [,c]	All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x and y must both be arrays or both be single variables /constants otherwise an error will be generated. c can be either an arrays or single variable/constant. See the Micromite User manual for full details of parameter usage.

PLAY FLAC file\$ [, interrupt]	Will play a FLAC file on the I2S output. Supported frequencies are: 44100Hz 16-bit(CD quality) and 24-bit 48000Hz 16-bit and 24-bit 88200Hz 16-bit and 24-bit 96000Hz 24-bit 192000Hz 24-bit (252MHz only) 'file\$' is the FLAC file to play (the extension of .flac will be appended if missing). The FLAC file is played in the background. 'interrupt' is optional and is the name of a subroutine which will be called when the file has finished playing.
PRINT #GPS, string\$	Outputs a NMEA string to an opened GPS device. The string must start with a \$ character and end with a * character. The Checksum is calculated automatically by the firmware and is appended to the string together with cr,lf
PWM 2, freq, 2A [,2B] [,2C]	See description of the PWM command in the Micromite User Manual. This command allows the specification of a frequency for a third PWM channel
RBOX x1, y1, w, h [, r] [,c] [,fill]	All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x1, y1, w, and h must all be arrays or all be single variables /constants otherwise an error will be generated. r, c, and fill can be either arrays or single variables/constants. See the Micromite User manual for full details of parameter usage.
SPRITE	The BLIT and SPRITE commands can be used interchangeably. The command is only available on VGA, SSD1963 (pin RD must be connected and specified) and ILI9341 (SPI-IN must be connected) displays
SPRITE x1, y1, x2, y2, w, h	Copies the memory area specified by top right coordinate x1, y1 and of width w and height h to a new location where the top right coordinate is x2, y2.
SPRITE CLOSE [#]n	Closes sprite n and releases all memory resources. Updates the screen (see SPRITE HIDE). Sprites which have been “copied” cannot be closed until all “copies” have been closed
SPRITE CLOSE ALL	Closes all sprites and releases all memory resources. It does not change the screen.
SPRITE COPY [#]n, [#]m, nbr	Makes a copy of sprite “n” to “nbr” of new sprites starting a number “m”. Copied sprites share the same loaded image as the original to save memory
SPRITE HIDE [#]n	Removes sprite n from the display and replaces the stored background. To restore a screen to a previous state sprites should be hidden in the opposite order to which they were written "LIFO"
SPRITE INTERRUPT sub	Specifies the name of the subroutine that will be called when a sprite collision occurs. See Appendix C for how to use the function SPRITE to interrogate details of what has collided

SPRITE LOAD [#]n, fname\$ [,colour]	Loads the file fname\$ as a sprite into buffer number n. The file must be in PNG format RGB888 or RGBA888. If the file extension .PNG is omitted then it will be automatically added. The parameter “colour” specifies the background colour for the sprite. Pixels in the background colour will not overwrite the background when the sprite is displayed. Colour defaults to zero
SPRITE MOVE	Actions a single atomic transaction that re-locates all sprites which have previously had a location change set up using the SPRITE NEXT command. Collisions are detected once all sprites are moved and reported in the same way as from a scroll
SPRITE NEXT [#]n, x, y	Sets the X and Y coordinate of the sprite to be used when the screen is next scrolled or the SPRITE MOVE command is executed. Using SPRITE NEXT rather than SPRITE SHOW allows multiple sprites to be moved as part of the same atomic transaction.
SPRITE NOINTERRUPT	Disables collision interrupts
SPRITE READ [#]n, x, y, w, h	Reads the display area specified by coordinates x and y, width w and height h into buffer number n. If the buffer is already in use and the width and height of the new area are the same as the original then the new command will overwrite the stored area.
SPRITE SCROLLH n [,col]	Scrolls the background and any sprites on layer 0 n pixels to the right. n can be any number between -31 and 31. Sprites on any layer other than zero will remain fixed in position on the screen. By default the scroll wraps the image round. If “col” is specified the colour will replace the area behind the scrolled image
SPRITE SCROLLR x, y, w, h, delta_x, delta_y [,col]	Scrolls the region of the screen defined by top-right coordinates “x” and “y” and width and height “w” and “h” by “delta_x” pixels to the right and “delta_y” pixels up. By default the scroll wraps the background image round. If “col” is specified the colour will replace the area behind the scrolled image. Sprites on any layer other than zero will remain fixed in position on the screen. Sprites in layer zero where the centre of the sprite ($x + w/2$, $y + h/2$) falls within the scrolled region will move with the scroll and wrap round if the centre moves outside one of the boundaries of the scrolled region.
SPRITE SCROLLV n [,col]	Scrolls the background, and any sprites on layer 0, n pixels up. n can be any number between -MM.VRES-1 and MM.VRES-1. Sprites on any layer other than zero will remain fixed in position on the screen. . By default the scroll wraps the image round. If “col” is specified the colour will replace the area behind the scrolled image
SPRITE SHOW [#]n, x, layer, [orientation]	Displays sprite n on the screen with the top left at coordinates x, y. Sprites will only collide with other sprites on the same layer, layer zero, or with the screen edge. If a sprite is already displayed on the screen then the SPRITE SHOW command acts to move the sprite to the new location. The display background is stored as part of the command such that it can be replaced when the sprite is hidden or

SPRITE WRITE [#]n, x y	moved further. The orientation is an optional parameter, valid values are: 0 - normal display (default if omitted) 1 - mirrored left to right 2 - mirrored top to bottom 3 - rotated 180 degrees (= 1+2) Overwrites the display with the contents of sprite buffer n with the top left at coordinates x, y. SPRITE WRITE overwrites the complete area of the display. The background that is overwritten is not stored so SPRITE WRITE is inherently higher performing than SPRITE SHOW but with greater functional limitations.
SERVO 2 [, freq], 2A [,2B] [,2C]	See description of the SERVO command in the Micromite User Manual. This command allows the specification of a frequency for a third SERVO channel
SENSORFUSION type ax, ay, az, gx, gy, gz, mx, my, mz, pitch, roll, yaw [,p1] [,p2]	Calculates pitch, roll and yaw angles from accelerometer and magnetometer inputs. Valid fusion types are MAHONY and MADGWICK. Usage is described in Appendix A
SPI3 OPEN speed, mode, bits SPI READ nbr, array() SPI WRITE nbr, data1, data2, data3, ... etc or SPI WRITE nbr, string\$ or SPI WRITE nbr, array() SPI CLOSE	See Appendix D of the Micromite User Manual. Not available on the Micromite eXtreme64
TRIANGLE X1, Y1, X2, Y2, X3, Y3 [, C [, FILL]]	All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x1, y1, x2, y2, x3, and y3 must all be arrays or all be single variables /constants otherwise an error will be generated c and fill can be either arrays or single variables/constants. See the Micromite Plus manual for full details of parameter usage.
TTS [PHONETIC] "text" [,speed] [,pitch] [,mouth] [,throat] [, interrupt]	Outputs text as speech on the audio outputs. See http://www.retrobits.net/atari/sam.shtml for details of parameter usage. The speech is played in the background. 'interrupt' is optional and is the name of a subroutine which will be called when the speech has finished playing.

Functions (Micromite eXtreme Only)

BAUDRATE(comm [, timeout])	Returns the baudrate of any data received on one of the communications ports (1-4). Samples the port over the period in seconds specified by the timeout. The timeout defaults to one second if not specified. Returns zero if no activity on the port within the timeout period.
CURSOR(x)	Returns the current x coordinate (in pixels) of the cursor
CURSOR(y)	Returns the current y coordinate (in pixels) of the cursor
CURSOR(left)	Returns the current state of the left button (0 or 1)
CURSOR(right)	Returns the current state of the right button (0 or 1)
CURSOR(middle)	Returns the current state of the middle button (0 or 1)
GPS(ALTITUDE)	returns current altitude if sentence GGA enabled
GPS(DATE)	returns the normal date string corrected for local time e.g. "12-01-2017"
GPS(DOP)	returns DOP (dilution of precision) value if sentence GGA enabled
GPS(FIX)	returns 0=no fix, 1=fix, etc. if sentence GGA enabled
GPS(GEOID)	Returns the geoid-ellipsoid separation. if sentence GGA enabled
GPS(LATITUDE)	returns the latitude in degrees as a floating point number, values are -ve for South of equator
GPS LONGITUDE)	returns the longitude in degrees as a floating point number, values are -ve for West of the meridian
GPS(SATELLITES)	returns number of satellites in view if sentence GGA enabled
GPS(SPEED)	returns the ground speed in knots as a floating point number
GPS(TIME)	returns the normal time string corrected for local time e.g. "12:09:33"
GPS(TRACK)	returns the track over the ground (degrees true) as a floating point number
GPS(VALID)	returns: 0=invalid data, 1=valid data. ALWAYS CHECK THIS VALUE TO ENSURE DATA IS VALID BEFORE USING OTHER GPS() FUNCTION CALLS
SPRITE(...)	The function is available on VGA, SSD1963 (pin RD must be connected and specified) and ILI9341 (SPI-IN must be connected) displays. In addition there is an optimised version for 16-bit parallel ILI9341 displays (code ILI9341P16)

SPRITE(C, [#]n)	Returns the number of currently active collisions for sprite n. If n=0 then returns the number of sprites that have a currently active collision following a SPRITE SCROLL command
SPRITE(C, [#]n, m)	Returns the number of the sprite which caused the “m”th collision of sprite n. If n=0 then returns the sprite number of “m”th sprite that has a currently active collision following a SPRITE SCROLL command
SPRITE(H,[#]n)	Returns the height of sprite n. This function is active whether or not the sprite is currently displayed (active).
SPRITE(L, [#]n)	Returns the layer number of active sprites number n
SPRITE(N)	Returns the number of displayed (active) sprites
SPRITE(N,n)	Returns the number of displayed (active) sprites on layer n
SPRITE(S)	Returns the number of the sprite which last caused a collision. NB if the number returned is Zero then the collision is the result of a SPRITE SCROLL command and the SPRITE(C...) function should be used to find how many and which sprites collided.
SPRITE(W, [#]n)	Returns the width of sprite n. This function is active whether or not the sprite is currently displayed (active).
SPRITE(X, [#]n)	Returns the X-coordinate of sprite n. This function is only active when the sprite is currently displayed (active). Returns 10000 otherwise.
SPRITE(Y, [#]n)	Returns the Y-coordinate of sprite n. This function is only active when the sprite is currently displayed (active). Returns 10000 otherwise.
LCOMPARE(array1%(), array2%())	Compare the contents of two long string variables array1%() and array2%(). The returned is an integer and will be -1 if array1%() is less than array2%(). It will be zero if they are equal in length and content and +1 if array1%() is greater than array2%(). The comparison uses the ASCII character set and is case sensitive.
LGETSTR\$(array%(), start, length)	Returns part of a long string stored in array%() as a normal MMBasic string. The parameters start and length define the part of the string to be returned.
LINSTR(array%(), search\$ [,start])	Returns the position of a search string in a long string. The returned value is an integer and will be zero if the substring cannot be found. array%() is the string to be searched and must be a long string variable. Search\$ is the substring to look for and it must be a normal MMBasic string or expression (not a long string). The search is case sensitive. Normally the search will start at the first character in 'str' but the optional third parameter allows the start position of the search to be specified.
LLEN(array%())	Returns the length of a long string stored in array%()
MM.DEVICE\$	Returns “Micromite eXtreme, Microchip ID 0xhhhhhhhh”

OWSEARCH(pin, flag [,serial_number_mask])	Returns the onewire device serial number as an INTEGER. Flags are: 0 - Continue an existing search 1 - start a new search 4 - Continue an existing search for devices in the requested family 5 - start a new search for devices in the requested family (the MSB of the serial_number_mask) 8 - skip the current device family and return the next device 16 - verify that the device with the serial number in serial_number_mask is available
SPI3(n)	See Appendix D of the Micromite User Manual

Appendix A

Sensor Fusion

The Micromite eXtreme supports the calculation of pitch, roll and yaw angles from accelerometer and magnetometer inputs.

For information on this technology see <https://github.com/kriswiner/MPU-6050/wiki/Affordable-9-DoF-Sensor-Fusion>

The SENSORFUSION command supports both the MADGWICK and MAHONY fusion algorithms. The format of the command is:

SENSORFUSION type ax, ay, az, gx, gy, gz, mx, my, mz, pitch, roll, yaw [p1] [p2]

Type can be MAHONY or MADGWICK

Ax, ay, and az are the accelerations in the three directions and should be specified in units of standard gravitational acceleration.

Gx, gy, and gz are the instantaneous values of rotational speed which should be specified in radians per second.

Mx, my, and mz are the magnetic fields in the three directions and should be specified in nano-Tesla (nT)

Care must be taken to ensure that the x, y and z components are consistent between the three inputs. So , for example, using the MPU-9250 the correct input will be ax, ay,az, gx, gy, gz, **my, mx, -mz** based on the reading from the sensor.

Pitch, roll and yaw should be floating point variables and will contain the outputs from the sensor fusion.

The SENSORFUSION routine will automatically measure the time between consecutive calls and will use this in its internal calculations.

The Madwick algorithm takes an optional parameter p1. This is used as beta in the calculation. It defaults to 0.5 if not specified

The Mahony algorithm takes two optional parameters p1, and p2. These are used as Kp and Ki in the calculation. If not specified these default to 10.0 and 0.0 respectively.

A fully worked example of using the code is given on the BackShed forum at

http://www.thebackshed.com/forum/forum_posts.asp?TID=9321&PN=1&TPN=1

MMX144 GPIO Pinout

(viewed from component side)

2	40			1	1	
5	39			2	4	
22	38			3	21	
31	37			4	25	
39	36			5	36	
43	35			6	40	
45	34			7	44	
50	33			8	46	
52	32			9	51	(10K pull-up)
56	31			10	53	(10K pull-up)
60	30			11	59	
66	29			12	65	
68	28			13	67	
82	27			14	81	(10K pull-up)
84	26			15	83	(10K pull-up)
92	25			16	87	(10K pull-up)
101	24			17	100	(10K pull-up)
(Piezo) 103	23			18	102	(10K pull-up)
(Mouse) 135	22			19	113	(10K pull-up)
144	21			20	138	(Mouse)



Analogue Input

5v Tolerant



```

'Standard IO Mappings to Pins for MMX+144
'Legend for each is
'    A - analogue capable
'    5 - 5V tolerant
'    P - pulled up (use OC in your code when defining)
'
' use as follows
'
'    SETPIN(IO1) etc...

CONST IO1=1      'A
CONST IO2=4      'A
CONST IO3=21     'A
CONST IO5=36     'A
CONST IO6=40     'A
CONST IO7=44     'A
CONST IO8=46     '5
CONST IO9=51     '5P
CONST IO10=53    '5P
CONST IO11=59    'A
CONST IO12=65    'A
CONST IO13=67    'A
CONST IO14=81    '5P
CONST IO15=83    '5P
CONST IO16=87    '5P
CONST IO17=100   '5P
CONST IO18=102   '5P
CONST IO19=113   '5P
CONST IO20=128   '5
CONST IO21=144   'A
CONST IO22=135   '5
CONST IO23=103   '5
CONST IO24=101   '5
CONST IO25=92    '5
CONST IO26=84    '5
CONST IO27=82    '5
CONST IO28=68    '5
CONST IO29=66    'A
CONST IO30=60    'A
CONST IO31=56    'A
CONST IO32=52    '5
CONST IO33=50    'A
CONST IO34=45    '5
CONST IO35=43    'A
CONST IO36=39    'A
CONST IO37=31    'A
CONST IO38=22    'A
CONST IO39=5     'A
CONST IO40=2     'A

```

Appendix C – Sprites

See the SPRITE commands and functions for syntax details.

The concept of the sprite implementation is as follows:

1. Sprites are full colour and of any size. The collision boundary is the enclosing rectangle.
2. Sprites are loaded to a specific number (1-50)
3. Sprites are displayed using the SPRITE SHOW command
4. For each SHOW command the user must select a "layer". This can be between 0 and 10.
5. Sprites collide with sprites on the same layer, layer 0, or the screen edge
6. Layer 0 is a special case and sprites on all other layers will collide with it
7. The SCROLL commands leave sprites on all layers except layer 0 unmoved
8. Layer 0 sprites scroll with the background and this can cause collisions
9. There is no practical limit on the number of collisions caused by SHOW or SCROLL commands
10. The sprite function allows the user to fully interrogate the details of a collision
11. A SHOW command will overwrite the details of any previous collisions for that sprite
12. A SCROLL command will overwrite details of previous collisions for ALL sprites
13. To restore a screen to a previous state sprites should be removed in the opposite order to which they were written "LIFO"

Because moving a sprite or, particularly, scrolling the background can cause multiple sprite collisions it is important to understand how they can be interrogated.

The best way to deal with a sprite collision is using the interrupt facility. A collision interrupt routine is set up using the SPRITE INTERRUPT command.

e.g. SPRITE INTERRUPT collision

The following is a pro-forma for identifying all collisions that have resulted from either a SPRITE SHOW command or a SCROLL command

```
,
' This routine demonstrates a complete interrogation of collisions
,
sub collision
  local integer i
' First use the SPRITE(S) function to see what caused the interrupt
if sprite(S) <> 0 then 'collision of specific individual sprite
  'sprite(S) returns the sprite that moved to cause the collision
  print "Collision on sprite ",sprite(S)
  process_collision(sprite(S))
  print ""
,
  else '0 means collision of one or more sprites caused by background move
    ' SPRITE(C, 0) will tell us how many sprites had a collision
    print "Scroll caused a total of ",sprite(C,0)," sprites to have
collisions"
```

```

    for i=1 to sprite(C,0)
        ' SPRITE(C, 0, i) will tell us the sprite number of the "I"th sprite
        print "Sprite ",sprite(C,0,i)
        process_collision(sprite(C,0,i))
    next i
    print ""
endif
end sub
' get details of the specific collisions for a given sprite
sub process_collision(S as integer)
    local integer i ,j
    'sprite(C, #n) returns the number of current collisions for sprite n
    print "Total of ",sprite(C,S)," collisions"
    for i=1 to sprite(C,S)
        ' SPRITE(C, S, i) will tell us the sprite number of the "I"th sprite
        j=sprite(C,S,i)
        if j=100 then
            print "collision with left of screen"
        else if j=101 then
            print "collision with top of screen"
        else if j=102 then
            print "collision with right of screen"
        else if j=103 then
            print "collision with bottom of screen"
        else
            print "Collision with sprite ",sprite(C,S,i) 'sprite(C, #n, #m)
returns details of the mth collision
        endif
    next i
end sub

```