Micromite Plus

Advanced Features

MMBasic Ver 5.1

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

or http://mmbasic.com

The Micromite Plus is a new addition to the Micromite family which runs on 64 and 100-pin PIC32 microcontrollers. The Micromite Plus implements all the features of the standard Micromite as described in the Micromite User Manual. It also has many 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 Plus. For general Micromite programming you should refer to the Micromite User Manual in addition to this manual.

Contents

Introduction

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

The Micromite Plus is an extension of the standard Micromite; all features of the Micromite are also supported in the Micromite Plus. 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 Plus.

The following summarises the new features in the Micromite Plus as compared to the standard Micromite:

MX470 Processor

The Micromite Plus is based on the Microchip PIC32MX470 32 bit microcontroller. This chip is available in 64 and 100-pin surface mount packages and is two to three times faster than the MX170 chip used in the standard Micromite. In BASIC the space available for both programs and variables is doubled to approx 100KB flash and 100KB RAM.

The Micromite Plus uses a crystal for timekeeping. This allows the Micromite Plus to support USB communications and also ensures that its internal clocks are much more accurate.

I/O Pins

The 64-pin Micromite Plus has up to 45 free I/O pins and the 100-pin chip 77. Of these 28 pins (on either chip) can be analog inputs.

The Micromite Plus has two SPI ports and up to four serial COM ports. All serial COM ports are high speed (over 1,000,000 baud). The provision of two SPI ports means that one can be used for SPI LCD's, touch and SD card interfaces leaving the other entirely free for use in a BASIC program. If the SPI is not used for these features then both SPI interfaces can be used in BASIC programs.

USB

The Micromite Plus has a built in USB 2.0 interface. The USB is integrated in the chip and no other hardware or components are required. MMBasic uses the USB CDC protocol which allows the USB interface to be used as the console with a host computer running a terminal emulator such as Tera Term.

The USB console operates in parallel with the serial console, anything received from either of the inputs is placed in the input queue for the interpreter or the program to read and anything outputted by the program or interpreter will be sent to either devices.

The USB feature is optional, if nothing is connected to the USB interface MMBasic will ignore it.

SD Card

The Micromite Plus includes full support for SD cards. This includes opening files for reading, writing or random access and loading and saving programs. The firmware will work with cards up to 64GB formatted in FAT16 or FAT32 and the files created can also be read and written on personal computers running Windows, Linux or the Mac operating system.

Up to 5 files can be open simultaneously. Using the OPEN command files can be opened and read from using INPUT, LINE INPUT, or INPUT$() or written to using PRINT or WRITE with. A program can be saved using the SAVE command and reloaded or run using LOAD. Programs and files stored on the card can be listed with the FILES command and deleted using the KILL command. The current working directory can be changed using CHDIR and a new directory be created with MKDIR.

The LOAD IMAGE command can be used to load a bitmap image from the SD card and display it on an attached LCD display panel. This can be used to draw a logo or add a background on the display.

The SD card feature is entirely optional and MMBasic will operate normally if it is not used.

SSD1963 Based LCD Display Panels

In addition to supporting LCD's that use SPI controllers (ie, the ILI9341 and ST7735) the Micromite Plus will also work with colour LCD displays that use the SSD1963 controller. These are available in sizes from 4.3 inch to 7 inch for prices that range from US$30 to US$50 on eBay.

SSD1963 based LCD panels use a parallel interface so the Micromite Plus can transfer data much faster than via SPI resulting in a very quick screen update. These displays are also much larger, have more pixels and are brighter. MMBasic will drive them using 24-bit true colour for a full colour rendition (16 million colours).

All LCD panels that use the SSD1963 controller have a SD card socket and touch sensitive screen, both of which are fully supported by MMBasic on the Micromite Plus.

The LCD display panel feature is entirely optional and if not configured MMBasic will operate normally.

Additional LCD Panel Features

The Micromite Plus includes six fonts that are built in and provides for up to ten additional fonts that can be embedded in the program or dynamically loaded from the SD card under program control. It also provides a set of read only variables to return the coordinates of the next print position on the s.

The Micromite Plus provides an optional PWM (Pulse Width Modulated) output for controlling the brightness of the LCD backlight. The Micromite Plus will also work with SSD1963 based LCD panels that are configured to use their controller to control the backlight.

Touch Input

The Micromite Plus extends the standard Micromite touch sensitive features with two new capabilities:

- Functions to detect if the touch is down or up, to return reference of the control currently being touched (see "Advanced Graphics" below) and also the last position or control touched.
- The ability to drive a connected Piezo buzzer to produce a click when a screen control is touched.

Advanced Graphics

The Micromite Plus incorporates a suite of advanced graphic controls that respond to touch, these include on screen switches, buttons, indicator lights, keyboard, etc. MMBasic will draw the control and animate it (ie, a switch will depress when touched). All that the BASIC program needs to do is invoke a single line command to specify the basic details of the control.

Using these advanced graphic controls it is easy to create a sophisticated control panel that includes WISWIG screen elements that are familiar to uses of modern computers (eg, check boxes, radio buttons, etc). A Micromite Plus using these controls could be used to create a professional control panel for anything from industrial machinery to a simple reticulation controller.

Console Input/Output

You can attach a PS2 keyboard to the Micromite Plus and, if you are using a SSD1963 based LCD panel, display the output of the interpreter in the LCD. This turns the Micromite Plus into a completely self contained computer with its own keyboard, display and SD card for storage. Using the built in colour coded editor programs can be entered, edited and run without requiring another computer.

The keyboard interface supports the standard USA keyboard layout or special United Kingdom, French, German, Belgium, Italian or Spanish layouts.

Miscellaneous

Other features of the Micromite Plus include:

- The ability to disable the serial console if it is not required. This allows the two I/O pins previously used by the console to be used as general I/O pins or as a fourth serial port (COM4:).
- The CPU command can vary the processor speed from 30MHz to 120MHz. The CPU command can also put the processor to sleep.
- The ability to specify the I/O pins used for connecting to a real time clock. When this option is specified MMBasic will automatically get the correct time on power up.

Micromite Family Summary

	Micromite		Micromite Plus	
	28-pin Dual In Line	44-pin SMD	64-pin SMD	100-pin SMD
Maximum CPU Speed	48MHz	48MHz	120MHz	120MHz
Maximum BASIC Program Size	59KB	59KB	100KB	100KB
RAM Memory Size (used for variables, buffers, etc)	52KB	52KB	108KB	108KB
Clock Speed (MHz)	5 to 48	5 to 48	30 to 120	30 to 120
Serial Console for Programming and Control	✓	✓	✓	✓
USB Console (serial emulator over USB 2.0)			✓	✓
PS2 Keyboard and LCD Display Console			✓	✓
SD Card Interface (FAT16 or FAT32 up to 64GB)			✓	✓
Total Number of I/O pins	19	33	45	77
Number of Analog Inputs	10	13	28	28
Number of Serial I/O ports	2	2	3 or 4	3 or 4
Number of SPI Channels	1	1	2	2
Number of I^2C Channels	1	1	1 + RTC	1 + RTC
Maximum Number of 1-Wire I/O pins	19	33	45	77
Number of PWM or Servo Channels	5	5	5	5
Supports 1.8", 2.4" and 2.8" SPI TFT LCD Displays	✓	✓	✓	✓
Supports 4.3", 5" and 7" TFT LCD Displays			✓	✓
Supports Resistive Touch Panels	✓	✓	✓	✓
Power Requirements	3.3V 30mA	3.3V 30mA	3.3V 80mA	3.3V 80mA

Suitable Microcontrollers

The microcontroller used in the Micromite Plus is available in two different frequency specifications (100MHz and 120MHz). The Micromite Plus will start up at 100MHz and if you have a 120MHz chip you can use the CPU command to step up to 120MHz. See http://microchip.com for the data sheets.

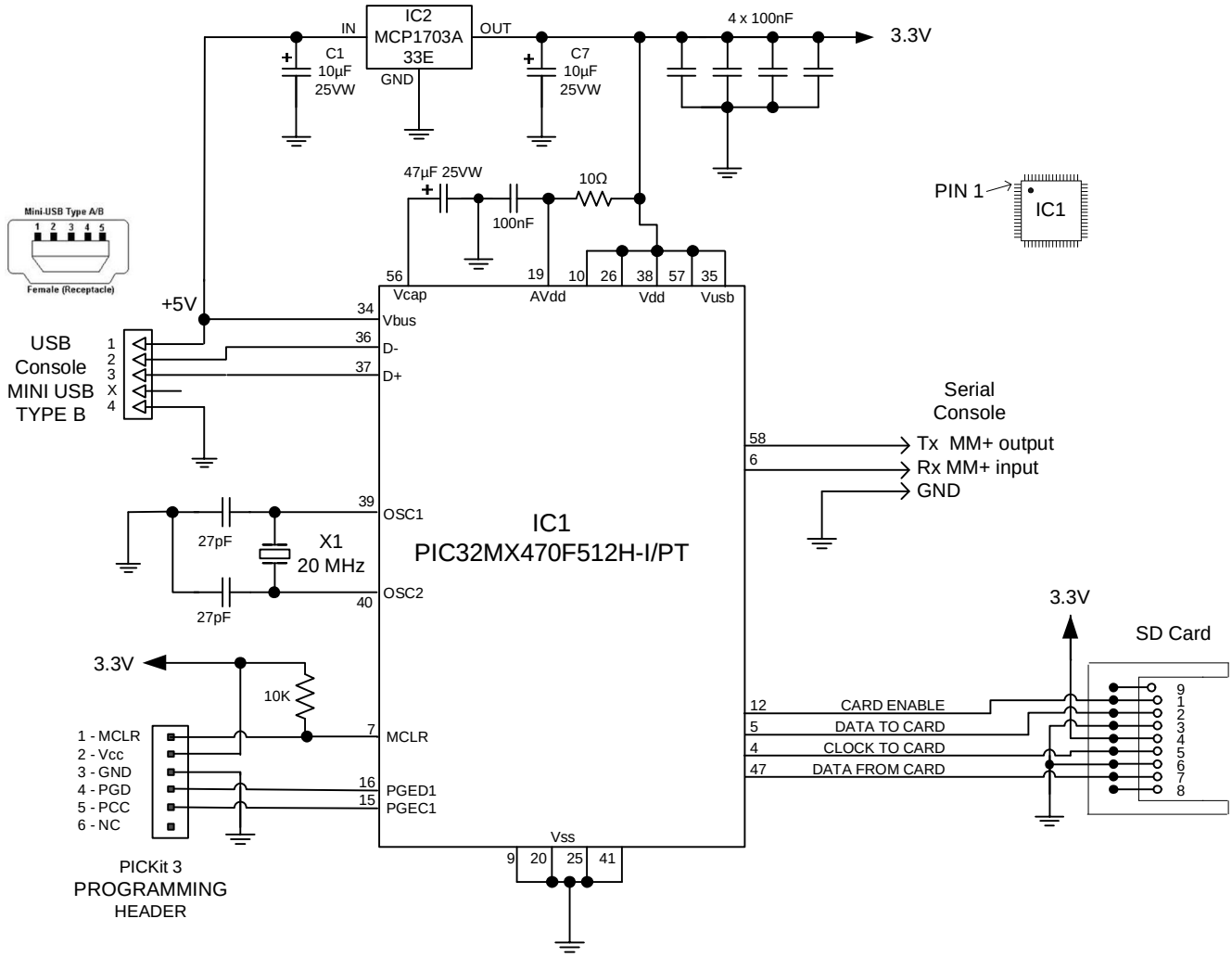
The recommended chips are:

PIC32MX470F512H-I/PT	64-pin TQFP package – maximum speed 100MHz
PIC32MX470F512H-120/PT	64-pin TQFP package – maximum speed 120MHz
PIC32MX470F512L-I/PF	100-pin TQFP package – maximum speed 100MHz
PIC32MX470F512L-120/PF	100-pin TQFP package – maximum speed 120MHz

All these chips are in a TQFP surface mount package with a lead pitch of 0.5mm. They are reasonably easy to solder and can be mounted on a carrier board (for example futurlec.com part code 64PINTQFP) or the Explore64 which is a breadboard compatible PCB (see http://geoffg.net/micromite.html for examples).

Typical Circuit

Note that the pin numbers refer to the 64-pin chip. For the 100-pin chip refer to the table in the next section.



Note that the only required components are:

- The 47µF capacitor on pin 56 (Vcap). This must be a multilayer ceramic or tantalum type.
- The 20MHz crystal and its two 27pF load capacitors.
- The 10KΩ pullup resistor on pin 7 (MCLR).

Micromite Plus Connections

The tick column ANA means ANALOG, DIG means digital I/O and 5V means a 5V tolerant pin. Pins marked SSD1963 xxx are required for a SSD1963 based display - see the "Micromite Advanced Features" manual for more details. Otherwise the notation is similar as described for the 28-pin version.

64-pin Micromite Plus

		ANA	DIG	5V		5V	DIG	ANA		
| DIGITAL INPUT ONLY	33		✓	✓		✓	✓		32	
USB +5V	*34*					✓	✓		31	
POWER (+2.3 to +3.6V)	*35*						✓	✓	30	
USB D-	*36*						✓	✓	29	COM1: ENABLE
USB D+	*37*						✓	✓	28	SSD1963 RESET
POWER (+2.3 to +3.6V)	*38*						✓	✓	27	SSD1963 RS (data/command)
20MHz CRYSTAL)	*39*								*26*	*POWER (+2.3 to +3.6V)*
20MHz CRYSTAL)	*40*								*25*	*GROUND*
GROUND	*41*						✓	✓	24	SSD1963 WR
PWM 1B	42		✓	✓			✓	✓	23	COUNT
I^2C DATA	43		✓	✓			✓	✓	22	
I^2C CLOCK	44		✓	✓			✓	✓	21	
SPI 1 IN (MISO)	45		✓	✓					*20*	*ANALOG GROUND*
	46		✓	✓					*19*	*ANALOG POWER (+2.3 to +3.6V)*
SPI 2 IN (MISO) | PWM 2B	47		✓				✓	✓	18	
PWM 1A	48		✓				✓	✓	17	COM3: Rx
COUNT	49	✓	✓				✓	✓	16	COM3: Tx
SPI 1 CLOCK	50	✓	✓				✓	✓	15	COM1: Tx
COUNT | WAKEUP | IR	51	✓	✓				✓	✓	14	
COUNT	52		✓	✓			✓	✓	13	COM2: Rx
PWM 2A	53		✓	✓			✓	✓	12	
| KEYBOARD CLOCK	54		✓	✓			✓	✓	11	COM2: Tx
| KEYBOARD DATA	55		✓	✓					*10*	*POWER (+2.3 to +3.6V)*
47µF TANT CAPACITOR (+)	*56*								*9*	*GROUND*
POWER (+2.3 to +3.6V)	*57*						✓	✓	*8*	SPI 1 OUT (MOSI)
COM4: Tx | CONSOLE Tx (OUT)	58		✓	✓					*7*	*RESET Normally 10KΩ to V+*
COM1: Rx	59		✓	✓			✓	✓	6	CONSOLE Rx (IN) | COM4: Rx
SSD1963 D0	60		✓	✓			✓	✓	5	SPI 2 OUT (MOSI) |
SSD1963 D1	61		✓	✓			✓	✓	4	SPI 2 CLOCK
SSD1963 D2	62	✓	✓				✓	✓	3	SSD1963 D7 |
PWM 1C | SSD1963 D3	63		✓	✓			✓	✓	2	SSD1963 D6 |
SSD1963 D4	64	✓	✓				✓	✓	1	SSD1963 D5 |

100-pin Micromite Plus

		ANA	DIG	5V		5V	DIG	ANA		
| DIGITAL INPUT ONLY	51		✓	✓		✓	✓		50	
	52		✓	✓		✓	✓		49	
	53		✓	✓		✓	✓		48	
USB +5V	*54*			✓		✓	✓		47	
POWER (+2.3 to +3.6V)	*55*								*46*	*POWER (+2.3 to +3.6V)*
USB D-	*56*								*45*	*GROUND*
USB D+	*57*						✓	✓	44	
	58		✓	✓			✓	✓	43	COM1: ENABLE
	59		✓	✓			✓	✓	42	SSD1963 RESET
	60		✓	✓			✓	✓	41	
	61		✓	✓		✓	✓		40	
POWER (+2.3 to +3.6V)	*62*					✓	✓		39	
20MHz CRYSTAL)	*63*					✓	✓		38	
20MHz CRYSTAL)	*64*								*37*	*POWER (+2.3 to +3.6V)*
GROUND	*65*								*36*	*GROUND*
I^2C CLOCK	66		✓	✓			✓	✓	35	
I^2C DATA	67		✓	✓			✓	✓	34	COUNT
PWM 1B	68		✓	✓			✓	✓	33	
	69		✓	✓			✓	✓	32	
SPI 1 CLOCK	70		✓	✓					*31*	*ANALOG GROUND*
SPI 1 IN (MISO)	71		✓	✓					*30*	*ANALOG POWER (+2.3 to +3.6V)*
SPI 1 OUT (MOSI)	72		✓	✓			✓		29	
	73		✓				✓		28	
PWM 1A	74		✓				✓	✓	27	
GROUND	*75*						✓	✓	26	COM3: Rx
COUNT	76	✓	✓				✓	✓	25	COM3: Tx
	77	✓	✓				✓	✓	24	COM1: Tx
COUNT | WAKEUP | IR	78	✓	✓				✓	✓	23	
PWM 1C	79		✓	✓			✓	✓	22	COM2: Rx
	80		✓	✓			✓	✓	21	
COUNT	81		✓	✓			✓	✓	20	COM2: Tx
PWM 2A	82		✓	✓		✓	✓		19	SSD1963 WR
KEYBOARD CLOCK	83		✓	✓		✓	✓		18	SSD1963 RS (data/command)
KEYBOARD DATA	84		✓	✓		✓	✓		17	
47µF TANT CAPACITOR (+)	*85*								*16*	*POWER (+2.3 to +3.6V)*
POWER (+2.3 to +3.6V)	*86*								*15*	*GROUND*
COM4: Tx | CONSOLE Tx (OUT)	87		✓	✓			✓	✓	14	
COM1: Rx	88		✓	✓		✓			*13*	*RESET Normally 10KΩ to V+*
COM4: Rx | CONSOLE Rx (IN)	89		✓	✓			✓	✓	12	SPI 2 OUT (MOSI)
	90		✓	✓			✓	✓	11	SPI 2 IN (MISO)
	91		✓	✓			✓	✓	10	SPI 2 CLOCK
	92		✓	✓		✓	✓		9	PWM 2B
SSD1963 D0	93		✓	✓		✓	✓		8	
SSD1963 D1	94		✓	✓		✓	✓		7	
	95		✓	✓		✓	✓		6	
	96		✓	✓			✓	✓	5	SSD1963 D7
	97		✓	✓			✓	✓	4	SSD1963 D6
SSD1963 D2	98	✓	✓				✓	✓	3	SSD1963 D5
SSD1963 D3	99		✓	✓					*2*	*POWER (+2.3 to +3.6V)*
SSD1963 D4	10	✓	✓			✓	✓		1	

Programming the Firmware

Programming the 64 and 100-pin Micromite Plus 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 Plus Pin Numbers	100-pin Micromite Plus Pin Numbers
	47µF Tantalum Capacitor to GND	56	85
1 - MCLR	Master Reset (active low)	7	13
2 - Vcc	Power Supply (3.3V)	10, 19, 26, 35, 38, and 57	2, 16, 30, 37, 46, 55, 62 and 86
3 - GND	Ground	9, 20, 25 and 41	15, 31, 36, 45, 65 and 75
4- PGD	Programming Data	16	25
5 - PGC	Programming Clock	15	24
6 - NC	Not used		

Notes:

- A pullup resistor of 10K is required between the MCLR pin and Vcc.
- A crystal is not required to program these chips and will be ignored if it is 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 the power to the chip being programmed.

SPI Based LCD Displays

The Micromite Plus includes support for three types of colour LCD display panels using a SPI interface. These are the:

- Standard 2.2”, 2.4" and 2.8” 240x320 pixel display with an ILI9341 controller as described in the Micromite User Manual.
- A 1.8" 128x160 pixel display with a ST7735 controller
- A 1.44" 128x128 pixel display using the ILI9163 controller.

On eBay you can find suitable displays by searching for the controller name (ILI9341, ST7735 or ILI9163).

ILI9341 Based Displays

The ILI9341 based displays are illustrated on the right.

Support for these displays is common across the Micromite family so, for more information, refer to the Micromite User Manual.

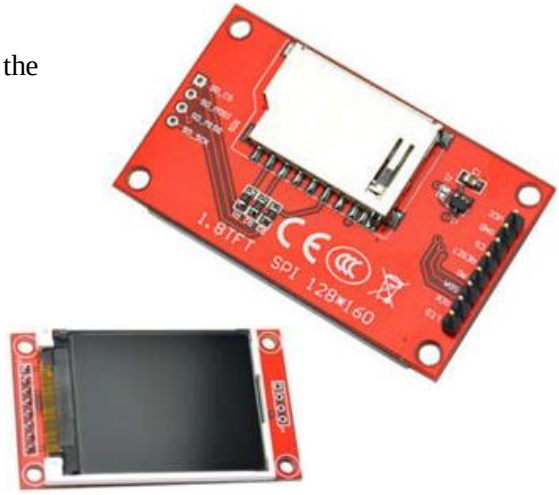
ST7735 Based Displays

The ST7735 based displays also use a SPI interface and are smaller than the ILI9341 based displays. ST7735 displays have the following basic specifications:

- A 1.8 inch display
- Resolution of 128 x 160 pixels and a colour depth of 262K/65K
- A ST7735 controller with a SPI serial interface

These do not come with a touch controller.

A typical ST7735 based display is illustrated on the right. On eBay you can find suitable displays by searching for the controller name (ST7735).



ILI9163 Based Displays

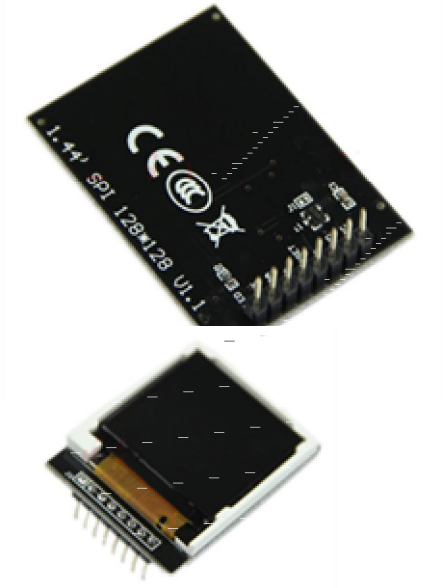
ILI9163 based displays have a 1.44 inch display with 128x128 pixels. They have an SPI interface and use the same connections as ST7735 based displays.

ILI9163 based displays use an SPI interface and have the following basic specifications:

- A 1.44 inch display
- Resolution of 128 x 128 pixels
- A ILI9163 controller with a SPI serial interface

A typical ILI9163 based display is illustrated on the right. On eBay you can find suitable displays by searching for the controller name (ILI9163).

Be warned, apparently some displays with a red PCB have a fault and will not work with the Micromite Plus. So, you should choose a display with a black PCB (as illustrated) as these have been tested and work correctly



Connecting SPI Based LCD Panels

The SPI based display controllers share the Micromite's the second SPI channel (SPI2) interface with the touch controller (if present) and the SD Card interface if implemented. If any of these features are enabled SPI2 will also be unavailable to BASIC programs (which can use the first SPI channel instead).

The following table lists the connections required between the ST7735 or ILI9163 based LCD display board and the Micromite Plus:

ST7735 or ILI9163 Display	Description	64-pin version	100-pin version
LED	Power supply for the backlight (see below)		
SCK	Display SPI Clock	Pin 4	Pin 10
SDA	Display Data In (MOSI)	Pin 5	Pin 12
A0	Display Data/Command Control	Configurable	
RESET	Display Reset (when pulled low)	Configurable	
CS	Display Chip Select	Optional - Configurable	
GND	Ground		
VCC	5V supply (the controller draws less than 10mA)		

Where a Micromite Plus connection is listed as "configurable" the specific pin should be specified with the OPTION LCDPANEL or OPTION TOUCH commands (see below).

The backlight power (the LED connection) should be supplied from the main 5V supply via a current limiting resistor. A typical value for this resistor is 39Ω for a current of about 30mA. The value of this resistor can be varied to reduce the power consumption or to provide a brighter display.

Important: Care must be taken with display panels that share the SPI port between a number of devices (display controller, touch, etc). In this case all the Chip Select signals must be configured in MMBasic or disabled by a permanent connection to 3.3V. If this is not done any unconnected Chip Select pins will float causing the wrong controller to respond to commands on the SPI bus.

Configuring an SPI Based LCD Panel

To use the display MMBasic must be configured using the OPTION LCDPANEL command which must be entered at the command prompt (not in a program).

The syntax is:

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

Where:

'controller' can be either ILI9341, ST7735 or ILI9163.

'orientation' can be LANDSCAPE, PORTRAIT, RLANDSCAPE or RPORTRAIT. These can be abbreviated to L, P, RL or RP. The R prefix indicates the reverse or "upside down" orientation.

'C/D pin' and 'reset pin' are the Micromite I/O pins to be used for these functions. Any free pin can be used.

'CS pin' can also be any I/O pin but is optional. If a touch controller is not used this parameter can be left off the command and the CS pin on the LCD display wired permanently to ground. If the touch controller is used this pin must then be specified and connected to a Micromite I/O pin.

To test the display you can enter the command GUI TEST LCDPANEL. You should see an animated display of colour circles being rapidly drawn on top of each other. Press any key on the console's keyboard to stop the test.

Important: The above test may not work if the display has a touch controller and the touch controller has not been configured (ie, the touch Chip Select pin is floating). In this case configure the touch controller (see below) and then retry GUI TEST LCDPANEL.

To verify the configuration you can use the command OPTION LIST to list all options that have been set including the configuration of the LCD panel.

SSD1963 Based LCD Displays

In addition to the SPI based controllers the Micromite Plus also supports colour LCD displays using the SSD1963 controller. These use a parallel interface, are available in sizes from 4.3" to 8" and have better specifications than the smaller displays. Typically they cost about US$30 for the 4.3" version to US$50 for the 7" version. All these displays have an SD card socket which is fully supported by MMBasic on the Micromite Plus.

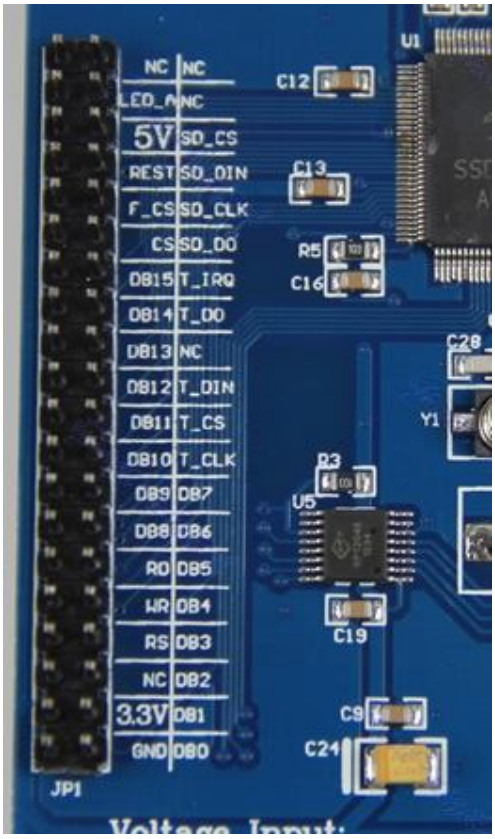
On eBay you can find suitable displays by searching for the controller name (SSD1963).

Because the SSD1963 controller uses a parallel interface the Micromite Plus can transfer data much faster than an SPI interface resulting in a very quick screen update. These displays are also much larger, have more pixels and are brighter. MMBasic drives them using 24-bit true colour for a full colour rendition (16 million colours).

The characteristics of these displays are:

- A 4.3, 5 or 7 inch display
- Resolution of 480 x 272 pixels (4.3" version) or 800 x 480 pixels (5” or 7" versions).
- A SSD1963 display controller with a parallel interface (8080 format).
- A touch controller (SPI interface).
- A full sized SD card socket.

There are a number of different designs using the SSD1963 controller but fortunately most Chinese suppliers have standardised on a single connector as illustrated below. It is strongly recommended that any display purchased has a matching connector – this provides some confidence that the manufacturer has followed the standard that the Micromite Plus is designed to use.



Connecting an SSD1963 Based LCD Panel

The SSD1963 controller uses a parallel interface while the touch controller and SD card use an SPI interface. The touch and SD card features are optional but if they are used they will use the second SPI port (SPI2).

The following table lists the connections required between the display board and the Micromite Plus to support the SSD1963 interface and the LCD display. The touch controller and SD card interfaces are listed further below.

SSD1963 Display	Description	64-pin Micromite+	100-pin Micromite+
DB0	Parallel Data Bus bit 0	Pin 60	Pin 93
DB1	Parallel Data Bus bit 1	Pin 61	Pin 94
DB2	Parallel Data Bus bit 2	Pin 62	Pin 98
DB3	Parallel Data Bus bit 3	Pin 63	Pin 99
DB4	Parallel Data Bus bit 4	Pin 64	Pin 100
DB5	Parallel Data Bus bit 5	Pin 1	Pin 3
DB6	Parallel Data Bus bit 6	Pin 2	Pin 4
DB7	Parallel Data Bus bit 7	Pin 3	Pin 5
CS	Chip Select (active low)	Ground (ie, always selected)	
WR	Write (active low)	Pin 24	Pin 19
RD	Read (active low)	Connect to 3.3V (ie, disabled)	
RS	Command/Data	Pin 27	Pin 18
RESET	Reset the SSD1963	Pin 28	Pin 42
5V	5V power for the backlight on some displays (most displays use the 3.3V supply for this).		
3.3V	Power supply.		
GND	Ground		

The following table lists the connections required to support the touch controller interface:

SSD1963 Display	Description	64-pin Micromite+	100-pin Micromite+
T_CS	Touch Chip Select	Configurable	
T_IRQ	Touch Interrupt	Configurable	
T_DIN	Touch Data In (MOSI)	Pin 5	Pin 12
T_CLK	Touch SPI Clock	Pin 4	Pin 10
T_DO	Touch Data Out (MISO)	Pin 47	Pin 11

The following table lists the connections required to support the SD card connector:

SSD1963 Display	Description	64-pin Micromite+	100-pin Micromite+
SD_CS	SD Card Chip Select	Configurable	
SD_DIN	SD Card Data In (MOSI)	Pin 5	Pin 12
SD_CLK	SD Card SPI Clock	Pin 4	Pin 10
SD_DO	SD Card Data Out (MISO)	Pin 47	Pin 11

Where a Micromite connection is listed as "configurable" the specific pin should be specified in the appropriate OPTION command (see below).

Most SSD1963 based LCD panels have three pairs of solder pads on the PCB which are grouped under the heading "Backlight Control". Normally the pair marked "LED-A" are shorted together with a zero ohm resistor and this allows control of the backlight's brightness with a PWM (pulse width modulated) signal on the LED-A pin of the display panel's main connector. The Micromite Plus fully supports this form of control and if you do not want to be bothered with the details you can specify the I/O pin used to drive the LED-A pin in the OPTION LCDPANEL command (below).

However, it is better to use the SSD1963 controller to generate this signal as it frees up one I/O pin and it allows the brightness to be controlled with a finer degree of resolution (1% instead of 5% steps). To use the SSD1963 for brightness control the zero ohm resistor should be removed from the pair marked "LED-A" and used to short the nearby pair of solder pads marked "1963-PWM". The Micromite Plus can then control the brightness by sending commands to the SSD1963 controller.

In either case the BACKLIGHT command will work the same and can be used to dynamically change the display's brightness.

Generally 7 inch and larger displays have a separate pin on the connector (marked 5V) for powering the backlight from a 5V supply. If this pin is not provided the backlight power will be drawn from the 3.3V pin. Note that the power drawn by the backlight can be considerable. For example, a 7 inch display will typically draw 330mA from the 5V pin.

Care must be taken with display panels that share the SPI port between a number of devices (SD card, touch, etc). In this case all the Chip Select signals must configured in MMBasic or disabled by a permanent connection to 3.3V. If this is not done the pin will float causing the wrong controller to respond to commands on the SPI bus.

On the Micromite Plus the second SPI channel (SPI2) is used to communicate with the touch controller and the SD Card interface. If any of these features are enabled SPI2 will be unavailable to BASIC programs (which can use the first SPI channel instead).

Configuring an SSD1963 Based LCD Panel

To use the display MMBasic must be configured using the OPTION LCDPANEL command which must be entered at the command prompt (not in a program).

The syntax is:

```
OPTION LCDPANEL controller, orientation [,LCD-A pin]
```

Where:

'controller' can be either:

- SSD1963_4 For a 4.3 inch display
- SSD1963_5 For a 5 inch display
- SSD1963_5A For an alternative version of the 5 inch display if SSD1963_5 does not work
- SSD1963_7 For a 7 inch display
- SSD1963_7A For an alternative version of the 7 inch display if SSD1963_7 does not work.

'orientation' can be LANDSCAPE, PORTRAIT, RLANDSCAPE or RPORTRAIT. These can be abbreviated to L, P, RL or RP. The R prefix indicates the reverse or "upside down" orientation.

'LCD-A pin' can also be any I/O pin but is optional. This can be used to control the brightness of the backlight if the internal SSD1963 controller is not used (see the notes above).

This command only needs to be run once. From then on MMBasic will automatically initialise the display on startup. If the LCD panel is no longer required the command OPTION LCDPANEL DISABLE can be used which will return the I/O pins for general use.

To verify the configuration you can use the command OPTION LIST to list all options that have been set including the configuration of the LCD panel.

To test the display you can enter the command GUI TEST LCDPANEL. You should see an animated display of colour circles being rapidly drawn on top of each other. Press any key on the console’s keyboard to stop the test.

Touch Support

The Micromite User Manual describes how to setup and calibrate a resistive touch sensitive panel. This section concentrates on the additional features offered by the Micromite Plus.

Configuring Touch

To configure the Micromite Plus for a touch panel the OPTION TOUCH command is used:

```
OPTION TOUCH T_CS pin, T_IRQ pin [, click pin]
```

Where:

'T_CS pin' and 'T_IRQ pin' are the Micromite I/O pins to be used for chip select and touch interrupt respectively (any free pins can be used).

'click pin' is exclusive to the Micromite Plus and specifies an I/O pin that will be driven briefly high when a screen control is touched. This can be used to drive a small Piezo buzzer which will produce a click sound providing an audible feedback when a GUI element on the screen is activated (see the section "Graphical User Interface Commands").

A typical buzzer that can be used is Altronics Part Number S6108.

Note that most I/O pins on the Micromite Plus are capable of driving 15mA (pins 50, 44 and 6 on the 64-pin chip can drive 30mA). However many buzzers require more that this so a transistor might be necessary to buffer the Micromite Plus output and drive the buzzer.

This command only needs to be run once as the parameters are stored in non volatile memory.

Calibrating the Touch Screen

Calibrating the touch screen is done using the GUI CALIBRATE command. When run this command will present a series of four targets on the screen that must be touched.

This process is fully described in the Micromite User Manual.

Touch Functions

To detect if and where the screen is touched you can use the following functions in a BASIC program.

Note that the first two functions are common across all versions of the Micromite, the remainder are unique to the Micromite Plus.

- TOUCH(X)
  Returns the X coordinate of the currently touched location or -1 if the screen is not being touched.
- TOUCH(Y)
  Returns the Y coordinate of the currently touched location or -1 if the screen is not being touched.
- TOUCH(DOWN)
  Returns true if the screen is currently being touched (this is much faster than TOUCH(X or Y)).
- TOUCH(UP)
  Returns true if the screen is currently NOT being touched (also faster than TOUCH(X or Y))
- TOUCH(LASTX)
  Returns the X coordinate of the last location that was touched.
- TOUCH(LASTY)
  Returns the Y coordinate of the last location that was touched.
- TOUCH(REF)
  Returns the reference number of the control that is currently being touched or zero if no control is being touched. See the section Advanced Graphics for more details.
- TOUCH(LASTREF)
  Returns the reference number of the control that was last touched.

The GUI BEEP Command

The Piezo buzzer specified in the OPTION TOUCH command can also be driven by a BASIC program using the command:

```
GUI BEEP msec
```

Where 'msec' is the number of milliseconds that the beeper should be driven. A time of 3mS produces a click while 100mS produces a short beep.

Touch Interrupts

On the Micromite Plus the GUI INTERRUPT command is used to setup a touch interrupt. The syntax is:

```
GUI INTERRUPT down [, up]
```

Where 'down' is the subroutine to call when a touch down has been detected. And optionally 'up' is the subroutine to call when the touch has been lifted from the screen ('up' and 'down' can point to the same subroutine if required).

For example, the following is equivalent to the above example:

```
GUI INTERRUPT MyInt
DO : LOOP

SUB MyInt
  PRINT TOUCH(X) TOUCH(Y)
END SUB
```

Specifying the number zero (single digit) as the argument will cancel both up and down interrupts. ie:

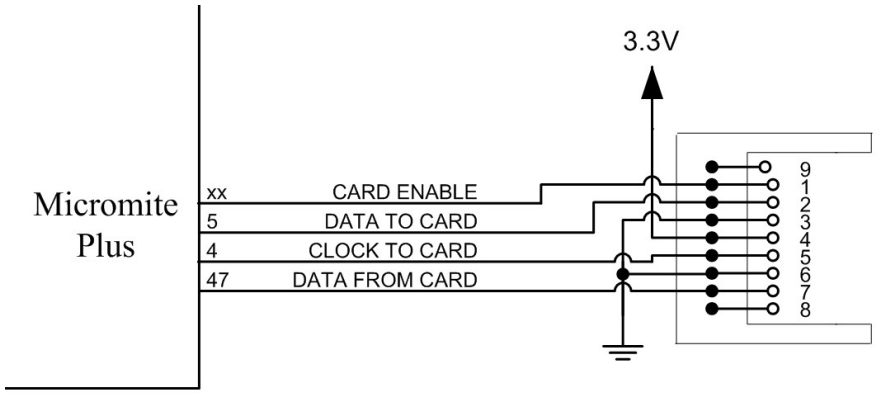
```
GUI INTERRUPT 0
```

SD Card Support

The Micromite Plus has full support for SD cards. This includes opening files for reading, writing or random access and loading and saving programs.

The firmware will work with cards up to 64GB formatted in FAT16 or FAT32 and the files created can also be read/written on personal computers running Windows, Linux or the Mac operating system.

The basic circuit diagram for connecting the SD card connector to the 64-pin Micromite Plus is illustrated below. Note that the pin number used for Card Enable is selected when configuring the interface.



Care must be taken with display panels that share the SPI port between a number of devices (SD card, touch, etc). In this case all the Chip Select signals must configured in MMBasic or alternatively disabled by a permanent connection to 3.3V. If this is not done any floating Chip Select signal lines will cause the wrong controller to respond to commands on the SPI bus.

Configuring the SD Card

To use the SD card MMBasic must be configured using the OPTION SDCARD command which must be entered at the command prompt (not in a program).

The syntax is:

```
OPTION SDCARD CS-pin [, CD-pin [,WP pin]]
```

Where:

'CS-pin' is the I/O pin number that will be used as chip select (pin 1 on the SD card connector).

'CD-pin' is optional and is the I/O pin number that will be used to connect to the card detect pin on the SD card connector. The Micromite will provide a weak pullup on this pin which is normally pulled high but, when a card is inserted, it will be connected to ground and the signal line pulled low. On some connectors this signal is not provided and in that case the Micromite Plus would need to be restarted if the SD card was changed.

'WP-pin' is optional and is the I/O pin number that will be used to connect to the write protect pin on the SD card connector. The Micromite will provide a weak pullup on this pin which is normally pulled high but, when a card is write protected, it will be connected to ground and the signal line pulled low.

Some SD card connectors reverse the polarity of the 'CD-pin' or 'WP-pin' signal - ie, they are open (and therefore the signal is high) when the card is inserted or write protected. In this case the pin numbers used for 'CD-pin' and 'WP-pin' can be a negative number. This will tell MMBasic to invert the polarity of these signals.

This command only needs to be run once. When the Micromite is restarted MMBasic will automatically initialise the SD card interface. If the SD card is no longer required the command OPTION SDCARD DISABLE can be used which will disable the SD card and return the I/O pins for general use.

To verify the configuration you can use the command OPTION LIST to list all options that have been set including the configuration of the SD Card.

Specific Examples

Micromite Plus only.

On the Explore 64 module (http://www.rictech.nz/pages/5/Products) the SD Card Chip Select (CS) signal is on pin 12 and the Card Detect (CD) signal is on pin 14. So, to enable the SD Card on this module the command is: OPTION SDCARD 12, 14

On the CGMICROBOARD2 module (www.circuitgizmos.com/products/cgmicroboard2/cgmicroboard2.shtml) the SD Card Chip Select (CS) signal is on pin 49 (there is no Card Detect). So, to enable the SD Card on this module the command is: OPTION SDCARD 49

The SnadPIC 100-pin PIC32 Module is useful if you are experimenting with the 100-pin Micromite Plus: http://www.microcontroller-board.com/snadpic-board-32/14-snadpic-pic32mx470f512l-development-board.html#/crystal_oscillator-20mhz/crystal_rtcc-no_without_crystal_r

This is the company's standard PIC32 experimental board modified to suit the Micromite Plus firmware (it has a 20MHz crystal and does not include a RTC crystal). If you are purchasing this board make sure that you specify these two details and the words "For Micromite" in the comments field.

On this module the SD Card Chip Select (CS) signal is on pin 14 and the Card Detect (CD) signal is on pin 18. So, to enable the SD Card on this module the command is: OPTION SDCARD 14, 18

Note that on the 100-pin Micromite Plus pin 18 is used for driving a SSD1963 based LCD panel and will conflict with its allocation as Card Detect. In this case the zero ohm resistor J3 on the underside of the SnadPIC board must be removed so that SD Card Detect is no longer on pin 18. Another pin may be selected via the CD pin on the top of the board.

MMBasic Support

MMBasic on the Micromite Plus supports the standard BASIC commands for working with storage systems.

Note that:

- All file/directory names must be in the 8.3 format (long names are not supported).
- Up to five files can be simultaneously open.
- In the following commands/functions the # in #fnbr is optional and may be omitted.

□ OPEN fname$ FOR mode AS #fnbr
Opens a file for reading or writing. 'fname$' is the file name in 8.3 format. 'mode' can be INPUT, OUTPUT, APPEND or RANDOM. '#fnbr' is the file number (1 to 10).

□ PRINT #fnbr, expression [[,;]expression] … etc
Outputs text to the file opened as #fnbr.

□ INPUT #fnbr, list of variables
Read a list of comma separated data into the variables specified from the file previously opened as #fnbr.

□ LINE INPUT #fnbr, variable$
Read a complete line into the string variable specified from the file previously opened as #fnbr.

□ CLOSE #fnbr [,#fnbr] …
Close the file(s) previously opened with the file number '#fnbr'.

Programs can be loaded from or saved to the SD card using two commands.

□ LOAD $fname [, R]
Load a BASIC program from the SD Card. The optional suffix ",R" will cause the program to be run after it has been loaded.

□ SAVE $fname
Save the current program to the SD card.

Basic file and directory manipulation can be done from within a BASIC program.

□ FILES [wildcard]
Search the current directory and list the files/directories found.

□ KILL $fname
Delete a file in the current directory.

□ MKDIR $dname
Make a sub directory in the current directory.

□ CHDIR $dname
Change into to the directory $dname. $dname can also be ".." (dot dot) for up one directory or "\" for the root directory.

□ RMDIR dir$
Remove, or delete, the directory 'dir$' on the SD card.

- SEEK #fnbr, pos
  Will position the read/write pointer in a file that has been opened for RANDOM access to the 'pos' byte.

Also there are a number of functions that support the above commands.

- INPUT$(nbr, #fnbr)
  Will return a string composed of ‘nbr’ characters read from a file previously opened for INPUT with the file number ‘#fnbr’. If less than ‘nbr’ characters are available the function will return with what it has (including an empty string if no characters are available).

- DIR$(fspec, type)
  Will search an SD card for files and return the names of entries found.

- EOF(#fnbr)
  Will return true if the file previously opened for INPUT with the file number ‘#fnbr’ is positioned at the end of the file.

- LOC(#fnbr)
  For a file opened as RANDOM this will return the current position of the read/write pointer in the file.

- LOF(#fnbr)
  Will return the current length of the file in bytes.

Load Image

The LOAD IMAGE command can be used to load a bitmap image from the SD card and display it on the attached LCD display panel. This can be used to draw a logo or add a background on the display.

The syntax of the command is:

```
LOAD IMAGE filename$ [, StartX, StartY]
```

Where ‘filename$’ is the image to load and ‘StartX’/’StartY’ are the coordinates of the top left corner of the image (these default to the top left corner of the display if not specified).

The image must be in BMP format and MMBasic will add “.BMP” to the file name if an extension is not specified. All types of the BMP format are supported including black and white and true colour 24-bit images.

The LOAD IMAGE can display full colour, full screen images which look gorgeous. With high resolution displays it will take some time to read the image from the SD Card (about 5 seconds for an 800x480 image).

Random File I/O

For random access the file should be opened with the keyword RANDOM. For example:

```
OPEN "filename" FOR RANDOM AS #1
```

To seek to a record within the file you would use the SEEK command which will position the read/write pointer to a specific byte. The first byte in a file is numbered one so, for example, the fifth record in a file that uses 64 byte records would start at byte 257. In that case you would use the following to point to it:

```
SEEK #1, 257
```

When reading from a random access file the INPUT$() function should be used as this will read a fixed number of bytes (ie, a complete record) from the file. For example, to read a record of 64 bytes you would use:

```
dat$ = INPUT$(64, #1)
```

When writing to the file a fixed record size should be used and this can be easily accomplished by adding sufficient padding characters (normally spaces) to the data to be written. For example:

```
PRINT #1, dat$ + SPACE$(64 – LEN(dat$);
```

The SPACE$() function is used to add enough spaces to ensure that the data written is an exact length (64bytes in this example). The semicolon at the end of the print command suppresses the addition of the carriage return and line feed characters which would make the record longer than intended.

Two other functions can help when using random file access. The LOC() function will return the current byte position of the read/write pointer and the LOF() function will return the total length of the file in bytes.

The following program demonstrates random file access. Using it you can append to the file (to add some data in the first place) then read/write records using random record numbers. The first record in the file is record number 1, the second is 2, etc.

```
RecLen = 64
OPEN "test.dat" FOR RANDOM AS #1
DO
   abort: PRINT
   PRINT "Number of records in the file =" LOF(#1)/RecLen
   INPUT "Command (r = read,w = write, a = append, q = quit): ", cmd$
   IF cmd$ = "q" THEN CLOSE #1 : END
   IF cmd$ = "a" THEN
      SEEK #1, LOF(#1) + 1
   ELSE
      INPUT "Record Number: ", nbr
      IF nbr < 1 or nbr > LOF(#1)/RecLen THEN PRINT "Invalid record" : GOTO abort
      SEEK #1, RecLen * (nbr - 1) + 1
   ENDIF
   IF cmd$ = "r" THEN
      PRINT "The record = " INPUT$(RecLen, #1)
   ELSE
      LINE INPUT "Enter the data to be written: ", dat$
      PRINT #1,dat$ + SPACE$(RecLen - LEN(dat$));
   ENDIF
LOOP
```

Random access can also be used on a normal text file. For example, this will print out a file backwards:

```
OPEN "file.txt" FOR RANDOM AS #1
FOR i = LOF(#1) TO 1 STEP -1
   SEEK #1, i
   PRINT INPUT$(1, #1);
NEXT i
CLOSE #1
```

Basic Drawing Features

The Micromite User Manual describes the basic drawing commands which are common across all versions of the Micromite. This section concentrates on the additional features offered by the Micromite Plus.

All coordinates and measurements on the screen are done in terms of pixels with the X coordinate being the horizontal position and Y the vertical position. The top left corner of the screen has the coordinates X=0 and Y=0 and the values increase as you move down and to the right of the screen.

Read Only Variables

In the Micromite Plus there are six read only variables which provide useful information about the display currently connected. The variables specific to the Micromite Plus are MM.HPOS and MM.VPOS, the others are supported by all Micromites:

- MM. HRES
  Returns the width of the display (the X axis) in pixels.
- MM. VRES
  Returns the height of the display (the Y axis) in pixels.
- MM.FONTHEIGHT
  Returns the height of the current font (in pixels). All characters in a font have the same height.
- MM.FONTWIDTH
  Returns the width of a character in the current font (in pixels). All characters in a font have the same width.
- MM.HPOS
  Returns the X coordinate of the text cursor (ie, the horizontal location (in pixels) of where the next character will be printed on the LCD panel)
- MM.VPOS
  Returns the Y coordinate of the text cursor (ie, the vertical location (in pixels) of where the next character will be printed on the LCD panel)

Colours

Colour is specified as a true colour 24 bit number where the top eight bits represent the intensity of the red colour, the middle eight bits the green intensity and the bottom eight bits the blue.

MMBasic will automatically translate all colours to the format required by the individual display controller which, in the case of the ILI9341, ST7735 and ILI9163 controllers, is 65K colours in the 565 format. In LCD panels using the SSD1963 controller colours are displayed using the full 24-bit colour range (16 million colours).

Fonts

The Micromite Plus has six built in fonts plus it can use embedded fonts or load additional fonts from the SD card.

The built in fonts are:

Font Number	Size (width x height)	Character Set	Description
1	8 x 13	All 95 characters	A small font where a dense display is required.
2	12 x 20	All 95 characters	General use on 480 x 272 displays
3	16 x 24	All 95 characters	General use on 800 x 480 displays
4	16 x 24 BOLD	All 95 characters	A bold version of font #3
5	24 x 32	All 95 characters	Large font, very clear
6	32 x 50	0 to 9, +, -, =, : and space	Numbers plus a few symbols. Very clear.

Embedded Fonts

Both the standard Micromite and the Micromite Plus support embedded fonts. These are fully described in the Micromite User Manual.

Loadable Fonts

The Micromite Plus can dynamically load fonts from an SD card into RAM as the program is run. To do this the LOAD FONT command is used:

```
LOAD FONT filename$ AS #nbr
```

Where filename$' is the font's file name as a string variable or string constant. '#nbr' is the font number to use (in the range of 7 to 16) when referencing the font. The hash (#) symbol is optional. Note that fonts loaded from an SD card are held into RAM and are cleared by resetting the Micromite Plus or running a new program.

Loadable fonts must be in the UTFT format which is normally used to specify the font in a C program and is therefore written in that language. However, MMBasic knows how to read these files and will extract the required font information from them.

The best source of fonts is Henning Karlsen's website at http://www.rinkydinkelectronics.com/r_fonts.php or you can use Google to search for UTFT Fonts.

There are many font files available covering a wide range of character sets as well as symbol fonts (Dingbats, etc) which are handy for creating on screen icons, etc. It is also possible to create your own font which can contain any symbols of any size as required.

Drawing Commands

The basic drawing commands are the same across the whole Micromite family. Refer to the Micromite User Manual for a description of these commands.

Backlight Control

The brightness of the backlight on a SSD1963 LCD panel can be controlled with the BACKLIGHT command:

```
BACKLIGHT percent
```

Where 'percent' is the degree of brightness ranging from 0 (fully off) to 100 (full brightness). This can be changed as often as required and makes a huge difference to the power requirements of the display. For example, a brightness of 50% will halve the current consumption (compared to 100%) while only making a small difference to the perceived visual brightness.

Load Image

As previously described in the "SD Card Support" section the LOAD IMAGE command can be used to load a bitmap image from the SD card and display it on the LCD display. This can be used to draw a logo or add an ornate background to the graphics drawn on the display.

Advanced Graphics

The Micromite Plus incorporates a suite of advanced graphic controls that respond to touch, these include on screen switches, buttons, indicator lights, keyboard, etc. MMBasic will draw the control and animate it (ie, a switch will depress when touched). All that the BASIC program needs to do is invoke a single line command to specify the basic details of the control.

Each control has a reference number called '#ref' in the description of the control. This can be any number between 1 and 100 and is used to reference a control. For example, a check box can be created thus:

```
GUI CHECKBOX #10, "Test", 100, 100, 50, rgb(BLUE)
```

And the program can check its value by using its reference number in the CtrlVal() function:

```
IF CtrlVal(#10) THEN ...
```

The # character is optional but serves to remind the programmer that this is not an ordinary number.

In the following commands any arguments that are in italic font (eg, *Width, Height*) are optional and if not specified will take the value of the previous command that did specify them. This means for example, that a number of radio buttons with the same size and colour can be specified with only the first button having to list all the details. Note that with the colour specification this is different to the Basic Drawing Commands which default to the last COLOUR command.

For all controls the font is not specified in these commands and will be whatever is set with the FONT command.

The advanced graphics controls are:

Frame

`GUI FRAME #ref, caption$, StartX, StartY, Width, Height, Colour`

This will draw a frame which is a box with round corners and a caption. A frame does not respond to touch but is useful when a group of controls need to be visually brought together. It can also used to surround a group of radio buttons and MMBasic will arrange for the radio buttons surrounded by the frame to be exclusive – that is, when one radio button is selected any other button that was selected and within the frame will be automatically deselected.

LED

`GUI LED #ref, caption$, CenterX, CenterY, Radius, Colour`

This will draw an indicator light (it looks like a panel mounted LED). When its value is set to a non zero number it will be illuminated and when it is set to zero it will be off (a dull version of its colour attribute). The caption will be drawn to the right of the LED and will use the colours set by the COLOUR command. A LED does not respond to touch.

Check Box

`GUI CHECKBOX #ref, caption$, StartX, StartY, Size, Colour`

This will draw a check box which is a small box with a caption. Both the height and width are specified with the 'Size' parameter. When touched an X will be drawn inside the box to indicate that this option has been selected and the control's value will be set to 1. When touched a second time the check mark will be removed and the control's value will be zero. The caption will be drawn to the right of the Check Box and will use the colours set by the COLOUR command.

Push Button

`GUI BUTTON #ref, caption$, StartX, StartY, Width, Height, FColour, BColour`

This will draw a momentary button which is a square switch with the caption on its face. When touched the visual image of the button will appear to be depressed and the control's value will be 1. When the touch is removed the value will revert to zero. Caption can be a single string with two captions separated by a | character (eg, "UP|DOWN"). When the button is up the first string will be used and when pressed the second will be used.

Switch

`GUI SWITCH #ref, caption$, StartX, StartY, Width, Height, FColour, BColour`

This will draw a latching switch with the caption on its face. When touched the visual image of the button will appear to be depressed and the control's value will be 1. When touched a second time the switch will be released and the value will revert to zero. Caption can be a single string with two captions separated by a | character (eg, "ON|OFF"). When this is used the switch will appear to be a toggle switch with each half of the caption used to label each half of the toggle switch.

Radio Button

`GUI RADIO #ref, caption$, CenterX, CenterY, Radius, Colour`

This will draw a radio button with a caption. When touched the centre of the button will be illuminated to indicate that this option has been selected and the control's value will be 1. When another radio button is selected the mark on this button will be removed and its value will be zero. Radio buttons are grouped together when surrounded by a frame and when one button in the group is selected all others in the group will be deselected. If a frame is not used all buttons on the screen will be grouped together.

The caption will be drawn to the right of the button and will use the colours set by the COLOUR command.

Display Box

`GUI DISPLAYBOX #ref, StartX, StartY, Width, Height, FColour, BColour`

This will draw a box with rounded corners. Any text can be displayed in the box by using the CtrlVal(r) = command. This is useful for displaying text, numbers and messages. This control does not respond to touch.

Text Box

`GUI TEXTBOX #ref, StartX, StartY, Width, Height, FColour, BColour`

This will draw a box with rounded corners. When the box is touched a QWERTY keyboard will appear on the screen. Using this virtual keyboard any text can be entered into the box including upper/lower case letters, numbers and any other characters in the ASCII character set. The new text will replace any text previously in the box.

The value of the control can be set to a string starting with two hash characters (##) and in that case the string (without the leading two hash characters) will be displayed in the box with reduced brightness. This can be used to give the user a hint as to what should be entered (called "ghost text"). Reading the value of the control displaying ghost text will return an empty string. When the control is used normally the ghost text will vanish.

MMBasic will try to position the virtual keyboard on the screen so as to not obscure the text box that caused it to appear. A pen down interrupt will be generated when the keyboard is deployed and a key up interrupt will be generated when the Enter key is touched and the keyboard is hidden.

Number Box

`GUI NUMBERBOX #ref, StartX, StartY, Width, Height, FColour, BColour`

This will draw a box with rounded corners. When the box is touched a numeric keypad will appear on the screen. Using this virtual keypad any number can be entered into the box including a floating point number in exponential format. The new number will replace the number previously in the box.

Similar to the Text Box the value of the control can set to a literal string with two leading hash characters (eg, "##Hint") and in that case the string (without the leading two characters) will be displayed in the box with reduced brightness. Reading this will return zero and when the control is used normally the ghost text will vanish.

MMBasic will try to position the virtual keypad on the screen so as to not obscure the number box that caused it to appear. A pen down interrupt will be generated when the keypad is deployed and a key up interrupt will be generated when the Enter key is touched and the keypad is hidden. Also, when the Enter key is touched the entered number will be evaluated as a number and the NUMBERBOX control redrawn to display this number.

Spin Box

```
GUI SPINBOX #ref, StartX, StartY, Width, Height, FColour, BColour, Step,
        Minimum, Maximum
```

This will draw a box with up/down icons on either end. When these icons are touched the number in the box will be incremented or decremented by the 'StepValue', holding down the touch will repeat the step at a fast rate. 'Minimum' and 'Maximum' set a limit on the value that can be entered. 'StepValue', 'Minimum' and 'Maximum' are optional and if not specified 'StepValue' will be 1 and there will be no limit on the number entered. A pen down interrupt will be generated every time up/down is touched or when automatic repeat occurs.

Caption

```
GUI CAPTION #ref, text$, StartX, StartY, Justify, FColour, BColour
```

This will draw a text string on the screen. 'Justify' is one or two letters where the first letter is the horizontal justification around X and can be L, C or R for LEFT, CENTER, RIGHT and the second letter is the vertical placement around Y and can be T, M or B for TOP, MIDDLE, BOTTOM. The default justification is left/top. This command is similar to the basic drawing command TEXT, the difference being that MMBasic will automatically dim this control if a keyboard or number pad is displayed.

If the colours are not specified this control will use the colours set by the COLOUR command.

Interacting with Controls

Using the following commands and functions the characteristics of the on screen controls can be changed and their value retrieved.

- = CTRLVAL(#ref)
  This is a function that will return the current value of a control. For controls like check boxes or switches it will be the number one (true) indicating that the control has been selected by the user or zero (false) if not. For controls that hold a number (eg, a SPINBOX) the value will be the number (normally a floating point number). For controls that hold a string (eg, TEXTBOX) the value will be a string. For example:
  ```
  PRINT "The number in the spin box is: " CTRLVAL(#10)
  ```

- CTRLVAL(#ref) =
  This command will set the value of a control. For off/on controls like check boxes it will override any touch input and can be used to depress/release switches, tick/untick check boxes, etc. A value of zero is off or unchecked and non zero will turn the control on. For a LED it will cause the LED to be illuminated or turned off. It can also be used to set the initial value of spin boxes, text boxes, etc. For example:
  ```
  CTRLVAL(#10) = 12.4
  ```

- GUI FCOLOUR colour, #ref1 [, #ref2, #ref3, etc]
  This will change the foreground colour of the specified controls to 'colour'. This is especially handy for a LED which can change colour.

- GUI BCOLOUR colour, #ref1 [, #ref2, #ref3, etc]
  This will change the background colour of the specified controls to 'colour'.

- = TOUCH(REF)
  This is a function that will return the reference number of the control currently being touched. If no control is currently being touched it will return zero.

- = TOUCH(LASTREF)
  This is a function that will return the reference number of the control that was last touched.

- GUI DISABLE #ref1 [, #ref2, #ref3, etc]
  This will disable the controls in the list. Disabled controls do not respond to touch and will be displayed dimmed. The keyword ALL can be used as the argument and that will disable all controls. For example:
  ```
  GUI DISABLE ALL
  ```

- GUI ENABLE #ref1 [, #ref2, #ref3, etc]
  This will undo the effects of GUI DISABLE and restore the controls in the list to normal operation. The keyword ALL can be used as the argument for all controls.

- GUI HIDE #ref1 [, #ref2, #ref3, etc]
  This will hide the controls in the list. Hidden controls will not respond to touch and will be replaced on the screen with the current background colour. The keyword ALL can be used as the argument.

- GUI SHOW #ref1 [, #ref2, #ref3, etc]
  This will undo the effects of GUI HIDE and restore the controls in the list to being visible and capable of normal operation. The keyword ALL can be used as the argument for all controls.

- GUI DELETE #ref1 [, #ref2, #ref3, etc]
  This will delete the controls in the list. This includes removing the image of the control from the screen using the current background colour and freeing the memory used by the control. The keyword ALL can be used as the argument and that will cause all controls to be deleted.

- GUI DEFAULT HIDDEN
  GUI DEFAULT SHOW
  The first command (GUI DEFAULT HIDDEN) will set the default state of newly created controls to hidden (ie, they will not be displayed on the screen when created). This is useful for defining controls that will later be made visible (using GUI SHOW) depending on the program logic. The GUI DEFAULT SHOW command can be used to restore the original behaviour.

Advanced Graphics Programming Techniques

When programming using the advance drawing commands implemented on the Micromite Plus there are a few techniques that make it easier to develop and maintain your program.

Use Constants for Control Reference Numbers

The advanced controls use a reference number to identify the control. To make it easy to read and maintain your program you should define these numbers as constants with easy to recognise names.

For example, in the following program fragment MAIN_SWITCH is defined as a constant and this constant is used wherever the reference number for that control is required:

```
CONST MAIN_SWITCH = 5
CONST ALARM_LED = 6
'…
GUI SWITCH MAIN_SWITCH, "ON|OFF", 330, 50, 140, 50,  RGB(white), RGB(blue)
GUI LED ALARM_LED, 215, 220,30, RGB(red)
'…
IF CtrlVal(MAIN_SWITCH) = 0 THEN …   ' for example turn the pump off
IF ALARM THEN CtrlVal(ALARM_LED) = 1
```

It is much easier remembering what MAIN_SWITCH does than remembering what control the number 5 refers to. Also, when you have a lot of controls it is much easier to renumber the controls when all their numbers are defined at the one place at the start of the program.

Use Interrupts and SELECT CASE Statements

Everything that happens on a screen using the advanced controls will be signalled by an interrupt, either touch down or touch up. So, if you want to do something when a control is changed, you should do it in an interrupt. Mostly you will be interested in when the touch (or pen) is down but in some cases you might also want to know when it is released.

Because the interrupt is triggered when the pen touches any control or part of the screen you need to discover what control was being touched. This is best performed using the TOUCH(REF) function and the SELECT CASE statement.

For example, in the following fragment the subroutine PenDown will be called when there is a touch and the function TOUCH(REF) will return the reference number of the control being touched. Using the SELECT CASE the alarm LED will be turned on or off depending on which button is touched. The action could be any number of things like raising an I/O pin to turn on a physical siren or printing a message on the console.

```
CONST ALARM_ON = 15
CONST ALARM_OFF = 16
CONST ALARM_LED = 33
GUI INTERRUPT PenDown
'…
GUI BUTTON ALARM_ON, "ALARM ON ", 330, 50, 140, 50, RGB(white), RGB(blue)
GUI BUTTON ALARM_OFF, "ALARM OFF ", 330, 150, 140, 50, RGB(white), RGB(blue)
GUI LED ALARM_LED, 215, 220, 30, RGB(red)
'…
DO : LOOP    ' the main program is doing something

' this sub is called when touch is detected
SUB PenDown
  SELECT CASE TOUCH(REF)
    CASE ALARM_ON
      CtrlVal(ALARM_LED) = 1
    CASE ALARM_OFF
      CtrlVal(ALARM_LED) = 0
  END SELECT
END SUB
```

The SELECT CASE can also test for other controls and perform whatever actions are required for them in their own section of the CASE statement.

The important point is that the maintenance of the controls (eg, responding to the buttons and turning the alarm LED off or on) is done automatically without the main program being involved – it can continue doing something useful like calculating some control response, etc.

Disable Invalid Controls

Sometimes a control will be invalid, for example an off button when the object of the control is not turned on. In that case your program should use the GUI DISABLE command to disable the control. This leaves the control still on the screen so that the user knows that it is there but it will be dimmed and will not respond to touch - which means that an interrupt will not be generated when it is touched.

Later, when the control becomes valid you can use the GUI ENABLE command to return it to use.

Hide Controls on Other Screens

Your program might need a number of screens with differing controls on each screen. This could be implemented by deleting the old controls and creating new ones when the screen is switched. But another way to do this is to create all the controls needed for all the screens and hide the controls that are not valid on the particular screen being displayed. When hidden a control will be completely erased from the screen and will not respond to touch. Therefore when switching screens it is a simple matter to hide the controls not wanted and show those that are.

If you use this technique you will probably want to create the controls in an initially hidden state. This can be achieved by using the GUI DEFAULT HIDDEN command in your program before the controls are created.

Using Basic Drawing Commands

There are two types of objects that can be on the screen. These are the GUI controls and the basic drawing objects (PIXEL, LINE, TEXT, etc). Mixing the two on the screen is not a good idea because MMBasic does not track the position of the basic drawing objects and they can clash with the GUI controls.

As a result, you should use either the GUI controls or the basic drawing objects – but not both.

If you are using GUI controls then everything on the screen must be a GUI control**.** So, for example, do not use TEXT but use GUI CAPTION instead. If you do this MMBasic will manage the screen for you including erasing and redrawing it as required.

Note that the CLS command (used to clear the screen) will automatically set any GUI controls on the screen to hidden (ie, it does a GUI HIDE ALL before clearing the screen).

Console Input/Output

In the Micromite all programming is done through the console. At the console you can enter commands, edit programs, run programs and observe the output of your program – including error messages!

The Micromite Plus has three console input/outputs. These are the serial console, serial emulation over USB and an optional PS2 keyboard and LCD display. All of these operate in parallel, anything received from any of the inputs is placed in the input queue for the interpreter or your program to read and anything outputted by your program or the interpreter will be sent to all three devices (if they are connected).

Serial Console

This is the traditional method of communicating with a Micromite and it is turned on by default when the Micromite firmware is programmed into the chip.

The default baud rate is 38400 and the performance of the editor and the speed of displaying text can be improved considerably by selecting a higher baud rate (see the OPTION BAUDRATE command in the "Micromite Manual").

If the serial console is not required it can be disabled. This allows the two I/O pins previously used by the console to be used as general I/O pins or as a fourth serial port (COM4:). The command to disable the serial console is:

```
OPTION CONSOLE OFF
```

If you want to re-enable the serial console you can do so with the command `OPTION CONSOLE ON`.

USB Console

See the heading "Typical Circuit" towards the start of this manual for details of the USB connections.

There is nothing that you need to do on the Micromite Plus to use the USB console. Just plug the USB cable from the Micromite Plus into your host computer and MMBasic will automatically create a virtual serial port over USB so that you can communicate with it from a Windows, Linux or Macintosh computer using nothing more than the USB port.

The communications protocol used is the CDC (Communication Device Class) protocol and there is native support for this in Windows 10, Linux (the cdc-acm driver) and Apple OS/X. Macintosh users can refer to the document "Using Serial Over USB on the Macintosh" on http://geoffg.net/maximite.html.

If you are using Windows you will need to install the Windows Serial Port Driver (available from http://geoffg.net/maximite.html). Full instructions are included in the download and when you have finished you should see the connection in Device Manager as a numbered communications port (eg, COM13).

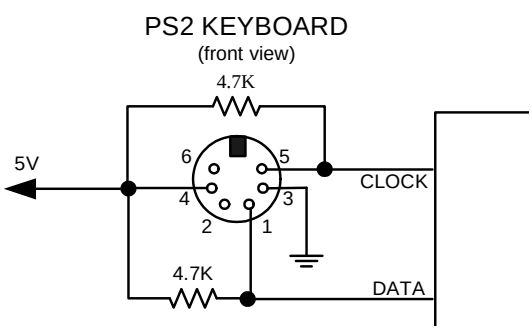
You can then use a terminal emulator such as Tera Term to connect to this communications port and it will work the same as if you were using a hardwired serial console. In Tera Term you do not have to specify a baud rate because the USB connection will run as fast as it can.

Be aware however that the USB connection will be reset if the Micromite Plus is reset and there are many things that can do this including the watchdog timer, the command CPU RESET and so on. For this reason the USB console is recommended only for situations where a stable program is running and a reset is not expected. For other situations it is recommended that the console be accessed via a USB to serial bridge connected to the serial console pins.

PS2 Keyboard

On the Micromite Plus you can attach a PS2 keyboard and, if you are using a SSD1963 based LCD panel, display the output of the interpreter on the LCD. This turns the Micromite Plus into a completely self contained computer with its own keyboard and display. Using the built in colour coded editor programs can be entered, edited and run without requiring another computer.

The connection diagram for the keyboard is shown on the right. The Micromite Plus enables weak pullups on the clock and data lines so the 4.7K resistors shown in the diagram are optional and for most keyboards there will be no ill effects if they are omitted.



Refer to the pinout diagrams in the “Micromite Manual” for the I/O pin numbers to use with the keyboard.

Before the keyboard can be used it must first be enabled by specifying the language of the keyboard:

```
OPTION KEYBOARD language
```

Where ‘language’ is a two character code such as US for the standard keyboard used in the USA, Australia and New Zealand. Other keyboard layouts that can be specified are United Kingdom (UK), French (FR), German (GR), Belgium (BE), Italian (IT) or Spanish (ES). Note that the non US layouts map some of the special keys present on these keyboards but the corresponding special character will not display as they are not part the standard Micromite fonts (another character will be used instead).

This command configures the I/O pins dedicated to the keyboard and initialises it for use. As with the similar commands for TOUCH, etc this option will be saved in flash memory and automatically applied on power up. If you want to remove the keyboard you can do this with the OPTION KEYBOARD DISABLE command.

LCD Display as the Console Output

The keyboard can be used on its own as an alternative input method but it works particularly well when the LCD display panel is used as the console output. The LCD must be one of the SSD1963 versions in the landscape or reverse landscape orientation and it must be first configured using OPTION LCDPANEL.

To enable the output to the LCD panel you should use the following command:

```
OPTION LCDPANEL CONSOLE [font [, fc [, bc [, blight]]]
```

'font' is the default font, 'fc' is the default foreground colour, 'bc' is the default background colour and 'blight' is the default backlight brightness (2 to 100). These settings are saved in flash and are used to configure MMBasic at power up. They are all optional and default to font 2, white, black and 100%.

Colour coding in the editor (see below) is also turned on by this command (OPTION COLOURCODE OFF will turn it off again). To disable using the LCD panel as the console the command is OPTION LCDPANEL NOCONSOLE.

Used with a PS2 keyboard this option turns the Micromite Plus into a self contained computer with its own keyboard and display. Rather like a modern version of the Maximite (see http://geoffg.net/maximite.html).

Miscellaneous Features

Real Time Clock

The I/O pins used for the Real Time Clock (RTC) can be set with the OPTION RTC command. This will define a private I2C interface to the RTC and free up the main I2C interface for other uses.

This command also causes MMBasic to automatically retrieve the time and date from the RTC on power up and set its internal clock accordingly. This can be useful when the Micromite Plus is being used as a general purpose computer with a keyboard and a SSD1963 display as the console.

Serial Interface

The Micromite Plus has built in support for up to four serial interfaces. COM1, COM2 and COM3 are available as standard and when the serial console is disabled (OPTION CONSOLE OFF) the I/O pins freed by this command can be opened as a fourth serial port (COM4).

All serial ports on the Micromite Plus can operate at high speed (up to 1M baud) at any CPU speed and support the INV, OC and S2 options. In addition COM1 supports the DE and 9BIT options (for RS485) as described in the Micromite User Manual.

SPI Interface

The Micromite Plus has built in support for two SPI interfaces. The commands to control these interfaces use the identifier SPI for the first SPI port and SPI2 for the second port. All commands and functions that can be used on the first port (SPI) as described in the Micromite User Manual can be used on the second by using the identifier SPI2. These are:

- SPI2 OPEN
- SPI2 WRITE
- SPI2 READ
- = SPI2(args, …)
- SPI2 CLOSE

SPI based displays, the touch controller and the SD Card interface (if implemented) will all use the second SPI interface (SPI2). If any of these features are enabled SPI2 will be unavailable to BASIC programs (which should use the first SPI channel instead).

Examples

Basic Explore 64 Configuration

To use the USB feature of the Explore 64 board you should check the requirements of your operating system (see the section *USB Console* above). Then it should be simply a case of plugging the Explorer 64 USB into your host computer, starting Tera Term or a similar terminal emulator and connecting to the virtual serial port created by the Micromite Plus on your computer.

To use the SD Card on the Explore 64 board you should run the following command at the command prompt:

```
OPTION SDCARD 12
```

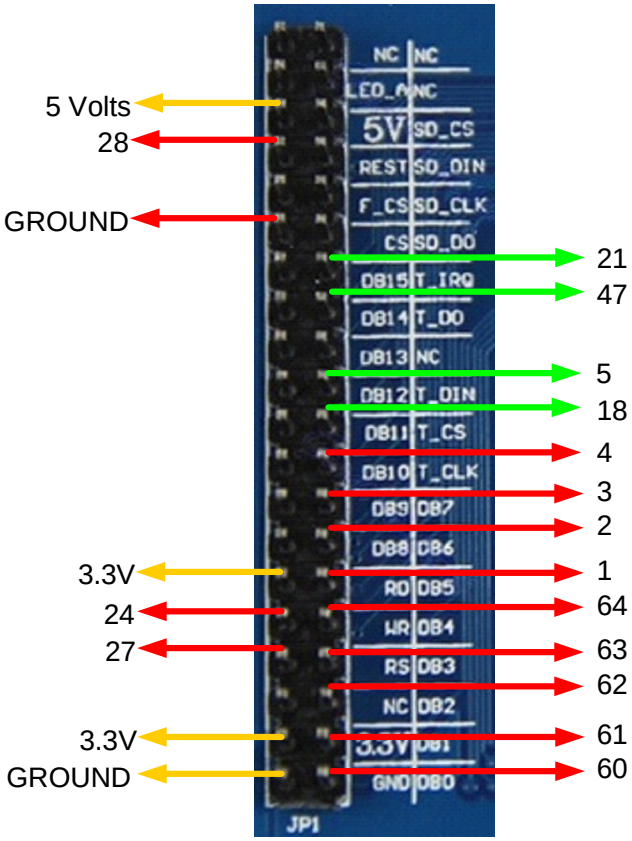
To test the SD command you can run the command FILES which should list the files on your SD card.

SSD1963 Based Display Connection

First, ensure that the jumper on the SSD1963 controller board is moved from the pair of solder pads marked LED-A to the pair marked 1963-PWM. This will place the backlight brightness under the control of SSD1963 chip (see "Connecting an SSD1963 Based LCD Panel" earlier in this manual).

If the board does not have these jumpers you will have to control the brightness via the LED-A pin as described in the above referenced section.

Next, connect the Explore 64 I/O pins to the SSD1963 display connector as shown in the diagram to the right. The power supply connections are shown in yellow, the SSD1963 controller connections are in red and the touch controller in green. Note that the touch Chip Select signal (pin 14) must be configured in MMBasic when using the SD card on the Explore 64 board. Without this the pin will float causing the touch controller to mistakenly respond to commands intended for the SD card.



You need to configure MMBasic for the display. An example is:

```
OPTION LCDPANEL SSD1963_7, L
```

This will configure the display as a 7" LCD panel in the landscape orientation. If your display is not the 7" version you should change the display type to `SSD1963_4` or `SSD1963_5` to suit your display.

To check that the LCD display panel is working correctly you can test it with the command:

```
GUI TEST LCDPANEL
```

You should see a rapid sequence of circles being drawn. Press any key to terminate the test.

Next, connect a pizeo buzzer from pin 50 of the Micromite Plus to ground (pin 50 has a 30mA drive capability and this should be enough to operate most pizeo buzzers). This will be used be used to generate a click sound when controls are touched, it is optional but the audible feedback makes using the advanced controls more intuitive.

The touch controller must be made known to MMBasic and the following gives an example:

```
OPTION TOUCH 18, 21, 50
```

This specifies that the touch controller's chip select is on pin 18, IRQ is on pin 21, and pin 50 is the click output.

To verify the configuration you can use the command OPTION LIST to list all options that have been set including the configuration of the LCD panel and touch input.

The touch controller then needs to be calibrated. To do this, enter this command and follow the on screen prompts which will request a touch on all four corners of the display:

```
GUI CALIBRATE
```

Finally, you can test the touch feature with the command:

```
GUI TEST TOUCH
```

When you touch the screen a white mark should appear exactly under the location touched. You can even draw on the screen to test your artistic skills !!

Keyboard and LCD Console

If you want to use the Micromite Plus and the SSD1963 display panel as a stand alone computer you can connect a keyboard and send the console output to the LCD panel.

To connect the keyboard follow the connections listed in the section "Console Input/Output". The PS2 clock signal (pin 5 on the DIN connector) should connect to pin 54 on the Micromite Explore 64 and the data signal (pin 1 on the DIN connector) should connect to pin 55. The keyboard should also be connected to +5V and ground.

Use the following command to configure the standard US layout and enable the keyboard:

```
OPTION KEYBOARD US
```

You can also direct the console output to the LCD panel with the command:

```
OPTION LCDPANEL CONSOLE
```

These options will be retained even when the power is removed so you can disconnect the serial or USB console and use the keyboard and LCD panel as a self contained computer programmed in the BASIC language (rather like the TRS-80 or Apple II of yesterday).

Example Program

As a test you can enter the following "Pump Control" demonstration program as shown in this YouTube video: https://youtu.be/BkJ2WbzO6dI. It will draw a selection of advanced controls on the screen that you can test.

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Demonstration program for the Micromite+
' It does not do anything useful except demo the various controls
'
' Geoff Graham, October 2015
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit
Dim ledsY
Colour RGB(white), RGB(black)

' reference numbers for the controls are defined as constants
Const c_head = 1, c_pmp = 2, sw_pmp = 3, c_flow = 4, tb_flow = 5
Const led_run = 6, led_alarm = 7
Const frm_alarm = 20, nbr_hi = 21, nbr_lo = 22, pb_test =23
Const c_hi = 24, c_lo = 25
Const frm_pump = 30, r_econ = 31, r_norm = 32, r_hi = 33
Const frm_log = 40, cb_enabled = 41, c_fname = 42, tb_fname = 43
Const c_log = 44, cb_flow = 45, cb_pwr = 46, cb_warn = 47
Const cb_alarm = 48, c_bright = 49, sb_bright = 50

' now draw the "Pump Control" display
CLS
GUI Interrupt TouchDown, TouchUp

' display the heading
Font 2,2 : GUI Caption c_head, "Pump Control", 10, 0
Font 3 : GUI Caption c_pmp, "Pump", 20, 60, , RGB(brown)

' now, define and display the controls
' first display the switch
Font 4
GUI Switch sw_pmp, "ON|OFF", 20, 90, 150, 50, RGB(white),RGB(brown)
CtrlVal(sw_pmp) = 1
```

```
' the flow rate display box
Font 3 : GUI Caption c_flow, "Flow Rate", 20, 170,, RGB(brown),0
Font 4 : GUI Displaybox tb_flow, 20, 200, 150, 45
CtrlVal(tb_flow) = "20.1"

' the radio buttons and their frame
Font 3 : GUI Frame frm_pump, "Power", 20, 290, 170, 163,RGB(200,20,255)
GUI Radio r_econ, "Economy", 35, 318, 25, RGB(230, 230, 255)
GUI Radio r_norm, "Normal", 35, 364
GUI Radio r_hi, "High", 35, 408
CtrlVal(r_norm) = 1     ' start with the "normal" button selected

' the alarm frame with two number boxes and a push button switch
Font 3 : GUI Frame frm_alarm, "Alarm", 220, 220, 200, 233,RGB(green)
GUI Caption c_hi, "High:", 232, 260, LT, RGB(yellow)
GUI Numberbox nbr_hi, 318,MM.VPos-6,90,MM.FontHeight+12,RGB(yellow),RGB(64,64,64)
GUI Caption c_lo, "Low:", 232, 325, LT, RGB(yellow),0
GUI Numberbox nbr_lo, 318,MM.VPos-6,90,MM.FontHeight+12,RGB(yellow),RGB(64,64,64)
GUI Button pb_test, "TEST", 257, 383, 130, 40,RGB(yellow), RGB(red)
CtrlVal(nbr_lo) = 15.7 : CtrlVal(nbr_hi) = 35.5

' draw the two LEDs
Const ledsX = 240, coff = 50    ' define their position
ledsY = 90 : GUI LED led_run, "Running", ledsX, ledsY, 30, RGB(green)
ledsY = ledsY+49 : GUI LED led_alarm, "Alarm", ledsX, ledsY, 30, RGB(red)
CtrlVal(led_run) = 1   ' the switch defaults to on so set the LED on

' the logging frame with check boxes and a text box
Colour RGB(cyan), 0
GUI Frame frm_log, "Log File", 450, 20, 330, 355, RGB(green),0
GUI Checkbox cb_enabled, "Logging Enabled", 470, 50, 30, RGB(cyan)
GUI Caption c_fname, "File Name", 470, 105
GUI Textbox tb_fname, 470, 135, 290, 40, RGB(cyan), RGB(64,64,64)
GUI Caption c_log, "Record:", 470, 205, , RGB(cyan), 0
GUI Checkbox cb_flow, "Flow Rate", 500, 245, 25
GUI Checkbox cb_alarm, "Alarms", 500, 285, 25
GUI Checkbox cb_warn, "Warnings", 500, 325, 25
CtrlVal(cb_enabled) = 1
CtrlVal(tb_fname) = "LOGFILE.TXT"

' define and display the spinbox for controlling the backlight
GUI Caption c_bright, "Backlight", 442, 415,,RGB(200,200,255),0
GUI Spinbox sb_bright, MM.HPos + 8, 400, 200, 50,,,10, 10, 100
CtrlVal(sb_bright) = 100

' All the controls have been defined and displayed. At this point
' the program could do some real work but because this is just a
' demo there is nothing to do.  So it just sits in a loop.
Do : Loop


' the interrupt routine for touch down
' using a select case command it has a different process for each
' control
Sub TouchDown
  Select Case Touch(REF)      ' find out the control touched
    Case cb_enabled           ' the enable check box
      If CtrlVal(cb_enabled) Then
        GUI Restore c_fname, tb_fname, c_log, cb_flow, cb_alarm, cb_warn
      Else
        GUI Disable c_fname, tb_fname, c_log, cb_flow, cb_alarm, cb_warn
      EndIf
    Case sb_bright            ' the brightness spin box
      BackLight CtrlVal(sb_bright)
    Case sw_pmp               ' the pump on/off switch
      CtrlVal(led_run) = CtrlVal(sw_pmp)
```

```
      CtrlVal(tb_flow) = Str$(CtrlVal(sw_pmp) * 20.1)
      CtrlVal(r_norm) = 1
    Case pb_test              ' the alarm test button
      CtrlVal(led_alarm) = 1
      GUI beep 250
    Case r_econ               ' the economy radio button
      CtrlVal(tb_flow) = Str$(CtrlVal(sw_pmp) * 18.3)
    Case r_norm               ' the normal radio button
      CtrlVal(tb_flow) = Str$(CtrlVal(sw_pmp) * 20.1)
    Case r_hi                ' the high radio button
      CtrlVal(tb_flow) = Str$(CtrlVal(sw_pmp) * 23.7)
  End Select
End Sub

' interrupt routine when the touch is removed
Sub TouchUp
  Select Case Touch(LASTREF)   ' use the last reference
    Case pb_test               ' was it the test button
      CtrlVal(led_alarm) = 0   ' turn off the LED
  End Select
End Sub
```

Read Only Variables (Micromite Plus Only)

MM.HPOS MM.VPOS	The current horizontal and vertical position (in pixels) following the last graphics or print command.
MM.ERRNO	Is set to the error number if a statement involving the SD card fails or zero if the operation succeeds. This is dependent on the setting of OPTION ERROR. The possible values for MM.ERRNO are: 0 = No error 1 = No SD card found 2 = SD card is write protected 3 = Not enough space 4 = All root directory entries are taken 5 = Invalid filename 6 = Cannot find file 7 = Cannot find directory 8 = File is read only 9 = Cannot open file 10 = Error reading from file 11 = Error writing to file 12 = Not a file 13 = Not a directory 14 = Directory not empty 15 = Hardware error accessing the storage media 16 = Flash memory write failure

Commands (Micromite Plus Only)

BACKLIGHT percent	Controls the brightness of the LCD panel's backlight (SSD1963 based display panels only). 'percent' is the degree of brightness ranging from 0 (fully off) to 100 (full brightness).
CHDIR dir$	Change the current working directory on the SD card to 'dir$' The special entry ".." represents the parent of the current directory and "." represents the current directory.
CLOSE [#]fnbr [,[#]fnbr] …	Close the file(s) previously opened with the file number '#fnbr'. The # is optional. Also see the OPEN command.
CTRLVAL(#ref) =	This command will set the value of an advanced control. '#ref' is the control's reference (a number from 1 to 100). For off/on controls like check boxes it will override any touch input and can be used to depress/release switches, tick/untick check boxes, etc. A value of zero is off or unchecked and non zero will turn the control on. For a LED it will cause the LED to be illuminated or turned off. It can also be used to set the initial value of spin boxes, text boxes, etc. For example: `CTRLVAL(#10) = 12.4`
FILES [fspec$]	Lists files in the current directory on the SD card. 'fspec$' (if specified) can contain search wildcards. Question marks (?) will match any character and an asterisk (*) will match any number of characters. If omitted, all files will be listed. For example: *.* Find all entries *.TXT Find all entries with an extension of TXT E*.* Find all entries starting with E X?X.* Find all three letter file names starting and ending with X
GUI CAPTION #ref, text$, startX, startY, justify, FColour, BColour	This will draw a text string on the screen. '#ref' is the control's reference (a number from 1 to 100). 'text$' is the string to display. 'startX' and 'startY' are the top left coordinates. 'justify' is one or two letters where the first letter is the horizontal justification around X and can be L, C or R for LEFT, CENTER, RIGHT and the second letter is the vertical placement around Y and can be T, M or B for TOP, MIDDLE, BOTTOM. The default justification is left/top. 'FColour and 'BColour' are RGB values for the foreground and background colours. 'justify', FColour and 'BColour' are optional and default to the colours set by the COLOUR command..
GUI SPINBOX #ref, startX, startY, width, height, FColour, BColour, Step, Minimum, Maximum	This will draw a box with up/down icons on either end. When these icons are touched the number in the box will be incremented or decremented. Holding down the up/down icons will repeat the step at a fast rate. '#ref' is the control's reference (a number from 1 to 100). 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls or set with the COLOUR command. 'Step' sets the amount to increment/decrement the number with each touch.

	'Minimum' and 'Maximum' set limits on the number that can be entered. All three parameters can be floating point numbers and are optional. The default for 'Step' is 1 and 'Minimum' and 'Maximum' if omitted will default to no limit.
GUI CHECKBOX #ref, caption$, startX, startY [, size] [, colour]	This will draw a check box which is a small box with a caption. When touched an X will be drawn inside the box to indicate that this option has been selected and the control's value will be set to 1. When touched a second time the check mark will be removed and the control's value will be zero. '#ref' is the control's reference (a number from 1 to 100). The string 'caption$' will be drawn to the right of the control using the colours set by the COLOUR command. 'startX' and 'startY' are the top left coordinates while 'size' set the height and width (the bix is square). 'colour' is an RGB value for the drawing colour. 'size' and 'colour' are optional and default to that used in previous controls.
GUI BCOLOUR colour, #ref1 [, #ref2, #ref3, etc]	This will change the background colour of the specified controls to 'colour' which is an RGB value for the drawing colour. '#ref' is the control's reference (a number from 1 to 100).
GUI BEEP msec	This will sound the pizeo buzzer if configured with the OPTION TOUCH command. 'msec' is the number of milliseconds that the buzzer should be driven. A time of 3mS produces a click while 100mS produces a short beep.
GUI BUTTON #ref, caption$, startX, startY, width, height [, FColour] [,BColour]	This will draw a momentary button which is a square switch with the caption on its face. When touched the visual image of the button will appear to be depressed and the control's value will be 1. When the touch is removed the value will revert to zero. #ref' is the control's reference (a number from 1 to 100). 'caption$' is the string to display on the face of the button. It can be a single string with two captions separated by a | character (eg, "UP|DOWN"). When the button is up the first string will be used and when pressed the second will be used. 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls or set with the COLOUR command.
GUI DEFAULT HIDDEN or GUI DEFAULT SHOW	This will set the default state of newly created controls to hidden (ie, they will not be displayed on the screen until GUI RESTORE is used). The SHOW setting can be used to restored the original behaviour.
GUI DELETE #ref1 [,#ref2, #ref3, etc] or GUI DELETE ALL	This will delete the controls in the list. This includes removing the image of the control from the screen using the current background colour and freeing the memory used by the control. '#ref' is the control's reference (a number from 1 to 100). The keyword ALL can be used as the argument and that will disable all controls.

GUI DISABLE #ref1 [,#ref2, #ref3, etc] or GUI DISABLE ALL	This will disable the controls in the list. Disabled controls do not respond to touch and will be displayed dimmed. '#ref' is the control's reference (a number from 1 to 100). The keyword ALL can be used as the argument and that will disable all controls. GUI ENABLE can be used to restore the controls.
GUI DISPLAYBOX #ref, startX, startY, width, height, FColour, BColour	This will draw a box with rounded corners that can be used to display a string '#ref' is the control's reference (a number from 1 to 100). 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls. Any text can be displayed in the box by using the CtrlVal(r) = command. This is useful for displaying text, numbers and messages. This control does not respond to touch.
GUI ENABLE #ref1 [,#ref2, #ref3, etc] or GUI ENABLE ALL	This will undo the effects of GUI DISABLE and restore the control(s) to normal operation. '#ref' is the control's reference (a number from 1 to 100). The keyword ALL can be used as the argument and that will disable all controls.
GUI FCOLOUR colour, #ref1 [, #ref2, #ref3, etc]	This will change the foreground colour of the specified controls to 'colour' which is an RGB value for the drawing colour. '#ref' is the control's reference (a number from 1 to 100).
GUI FRAME #ref, caption$, startX, startY, width, height, colour	This will draw a frame which is a box with round corners and a caption. '#ref' is the control's reference (a number from 1 to 100). 'caption$' is a string to display as the caption. 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. 'colour' is an RGB value for the drawing colour. 'width', 'height' and 'colour' are optional and default to that used in previous controls. A frame is useful when a group of controls need to be visually brought together. It is also used to surround a group of radio buttons and MMBasic will arrange for the radio buttons surrounded by the frame to be exclusive. ie, when one radio button is selected any other button that was selected and within the frame will be automatically deselected. A frame does not respond to touch.
GUI HIDE #ref1 [,#ref2, #ref3, etc] or GUI HIDE ALL	This will hide the controls in the list. Hidden controls do not respond to touch and will not be visible. '#ref' is the control's reference (a number from 1 to 100). The keyword ALL can be used as the argument and that will hide all controls. GUI SHOW can be used to restore the controls.
GUI INTERRUPT down [, up]	This command will setup an interrupt that will be triggered on a touch on the LCD panel and optionally if the touch is released. 'down' is the subroutine to call when a touch down has been detected. 'up' is the subroutine to call when the touch has been lifted from the screen ('up' and 'down' can point to the same subroutine if required). Specifying the number zero (single digit) as the argument will cancel both of these interrupts. ie: `GUI INTERRUPT 0`

GUI LED #ref, caption$, centerX, centerY, radius, colour	This will draw an indicator light which looks like a panel mounted LED. '#ref' is the control's reference (a number from 1 to 100). The string 'caption$' will be drawn to the right of the control using the colours set by the COLOUR command. 'centerX' and 'centerY' are the coordinates of the centre of the LED and 'radius' is the radius of the LED. 'colour' is an RGB value for the drawing colour. 'radius' and 'colour' are optional and default to that used in previous controls. When a LED's value is set to a non zero number it will be illuminated and when it is set to zero it will be off (a dull version of its colour attribute). The colour can be changed with the GUI FCOLOUR command. A LED does not respond to touch.
GUI NUMBERBOX #ref, startX, startY, width, height, FColour, BColour	This will draw a box with rounded corners that can be used to create a virtual numeric keypad for data entry. '#ref' is the control's reference (a number from 1 to 100). 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls. When the box is touched a numeric keypad will appear on the screen. Using this virtual keypad any number can be entered into the box including a floating point number in exponential format. The new number will replace the number previously in the box. The value of the control can set to a literal string (not an expression) starting with two hash characters. For example: `CtrlVal(nnn) = "##Enter Number"` and in that case the string (without the leading two hash characters) will be displayed in the box with reduced brightness. This can be used to give the user a hint as to what should be entered (called "ghost text"). Reading the value of the control displaying ghost text will return zero. When the control is used normally the ghost text will vanish. MMBasic will try to position the virtual keypad on the screen so as to not obscure the number box that caused it to appear. A pen down interrupt will be generated when the keypad is deployed and a key up interrupt will be generated when the Enter key is touched and the keypad is hidden. Also, when the Enter key is touched the entered number will be evaluated as a number and the NUMBERBOX control redrawn to display this number.
GUI RADIO #ref, caption$, centerX, centerY, radius, colour	This will draw a radio button with a caption. '#ref' is the control's reference (a number from 1 to 100). The string 'caption$' will be drawn to the right of the control using the colours set by the COLOUR command. 'centerX' and 'centerY' are the coordinates of the centre of the button and 'radius' is the radius of the button. 'colour' is an RGB value for the drawing colour. 'radius' and 'colour' are optional and default to that used in previous controls. When touched the centre of the button will be illuminated to indicate that this option has been selected and the control's value will be 1. When another radio button is selected the mark on this button will be removed and its value will be zero. Radio buttons are grouped together when surrounded by a frame and when one button in the group is selected all others in the group will be deselected. If a frame is not used all buttons on the screen will be grouped together.

GUI REDRAW #ref1 [,#ref2, #ref3, etc] or GUI REDRAW ALL	This will redraw the controls on the screen. It is useful if the screen image has somehow been corrupted. '#ref' is the control's reference (a number from 1 to 100). The keyword ALL can be used as the argument and that will disable all controls.
GUI SHOW #ref1 [,#ref2, #ref3, etc] or GUI SHOW ALL	This will undo the effects of GUI HIDE and restore the control(s) to being visible and capable of normal operation. '#ref' is the control's reference (a number from 1 to 100). The keyword ALL can be used as the argument and that will disable all controls.
GUI SPINBOX #ref, startX, startY, width, height, FColour, BColour, Step, Minimum, Maximum	This will draw a box with up/down icons on either end. When these icons are touched the number in the box will be incremented or decremented. Holding down the up/down icons will repeat the step at a fast rate. '#ref' is the control's reference (a number from 1 to 100). 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls. 'Step' sets the amount to increment/decrement the number with each touch. 'Minimum' and 'Maximum' set limits on the number that can be entered. All three parameters can be floating point numbers and are optional. The default for 'Step' is 1 and 'Minimum' and 'Maximum' if omitted will default to no limit.
GUI SWITCH #ref, caption$, startX, startY, width, height, FColour, BColour	This will draw a latching switch which is a square switch that latches when touched. '#ref' is the control's reference (a number from 1 to 100). 'caption$' is a string to display as the caption on the face of the switch. 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls. When touched the visual image of the button will appear to be depressed and the control's value will be 1. When touched a second time the switch will be released and the value will revert to zero. Caption can consist of two captions separated by a | character (eg, "ON|OFF"). When this is used the switch will appear to be a toggle switch with each half of the caption used to label each half of the toggle switch.
GUI TEXTBOX #ref, startX, startY, width, height, FColour, BColour	This will draw a box with rounded corners that can be used to create a virtual keyboard for data entry '#ref' is the control's reference (a number from 1 to 100). 'startX' and 'startY' are the top left coordinates while 'width' and 'height' set the dimensions. ' FColour and 'BColour' are RGB values for the foreground and background colours. 'width', 'height', FColour and 'BColour' are optional and default to that used in previous controls. When the box is touched a QWERTY keyboard will appear on the screen. Using this virtual keyboard any text can be entered into the box including upper/lower case letters, numbers and any other characters in the ASCII character set. The new text will replace any text previously in the box. The value of the control can set to a string starting with two hash characters. For example: `CtrlVal(nnn) = "##Enter Filename"`

	and in that case the string (without the leading two hash characters) will be displayed in the box with reduced brightness. This can be used to give the user a hint as to what should be entered (called "ghost text"). Reading the value of the control displaying ghost text will return an empty string. When the control is used normally the ghost text will vanish. MMBasic will try to position the virtual keyboard on the screen so as to not obscure the text box that caused it to appear. A pen down interrupt will be generated when the keyboard is deployed and a key up interrupt will be generated when the Enter key is touched and the keyboard is hidden.
INPUT #fnbr, list of variables	Same as the normal INPUT command except that the input is read from a file previously opened for INPUT as '#fnbr'. See the OPEN command.
KILL file$	Deletes the file specified by 'file$'. If there is an extension it must be specified.
LINE INPUT #fnbr, string-variable$	Same as the LINE INPUT command except that the input is read from a file previously opened for INPUT as '#fnbr'. See the OPEN command.
LOAD file$ [,R]	Loads a program called 'file$' from the current drive into program memory. If the optional suffix ,R is added the program will be immediately run without prompting. If an extension is not specified ".BAS" will be added to the file name.
LOAD FONT filename$ AS #nbr	This will load a font from the SD card and it can then be used in a similar manner to the built in fonts (ie, selected using the FONT command or specified in the TEXT command). 'filename$' is the font's file name as a string variable or string constant. '#nbr' is the font number to use (in the range of 7 to 16) when referencing the font. The hash (#) symbol is optional. All loaded fonts are cleared when a program is run or the editor is used. Also, a new font can be loaded using the same font number as a previously loaded font and will replace it. Loadable fonts must be in the UTFT format and must have the bitmap specified as a series of 8-pib numbers separated by commas. The best source of fonts is http://www.rinkydinkelectronics.com/r_fonts.php or you can use Google to search for UTFT Fonts.
LOAD IMAGE file$ [, x, y]	Load a bitmapped image from the SD card and display it on the LCD panel. "file$' is the name of the file and 'x' and 'y' are the screen coordinates for the top left hand corner of the image. If the coordinates are not specified the image will be drawn at the top left hand position on the screen. If an extension is not specified ".BMP" will be added to the file name. All types of the BMP format are supported including black and white and true colour 24-bit images.
MKDIR dir$	Make, or create, the directory 'dir$' on the SD card.
NAME old$ AS new$	Rename a file or a directory from 'old$' to 'new$'
OPEN fname$ FOR mode AS [#]fnbr	Opens a file for reading or writing. 'fname' is the filename (8 chars max) with an optional extension (3 chars max) separated by a dot (.). The default directory must be selected forst using CHDIR (ie, a path is not accepted). 'mode' is INPUT, OUTPUT, APPEND or RANDOM. INPUT will open the file for reading and throw an error if the file does not exist. OUTPUT will open the file for writing and will automatically overwrite any existing file with the same name. APPEND will also open the file for writing but it will not overwrite an existing file; instead any writes will be appended to the end of the file. If there is no existing file the APPEND mode will act the same as the OUTPUT mode (i.e. the file is created then opened for writing).

	RANDOM will open the file for both read and write and will allow random access using the SEEK command. When opened the read/write pointer is positioned at the end of the file. 'fnbr' is the file number (1 to 10). The # is optional. Up to 5 files can be open simultaneously. The INPUT, LINE INPUT, PRINT, WRITE and CLOSE commands as well as the EOF() and INPUT$() functions all use 'fnbr' to identify the file being operated on. See also OPTION ERROR and MM.ERRNO for error handling.
OPTION ERROR CONTINUE or OPTION ERROR ABORT	Set the treatment for errors in file input/output on the SD card. The option CONTINUE will cause MMBasic to ignore file related errors. The program must check the variable MM.ERRNO to determine if and what error has occurred. The option ABORT sets the normal behaviour (ie, stop the program and print an error message). The default is ABORT. Note that this option only relates to errors reading from or writing to the SD card:. It does not affect the handling of syntax and other program errors.
OPTION LCDPANEL controller, orientation, D/C pin, reset pin [,CS pin] or OPTION LCDPANEL controller, orientation [,LCD-A pin] or OPTION LCDPANEL DISABLE	Configures the Micromite and Micromite Plus to work with an attached LCD panel. 'controller' can be: • ILI9163 SPI based 1.44" panels using the ILI9163 controller • ST7735 SPI based 1.8" panels using the ST7735 controller • ILI9341 SPI based 2.2", 2.4" and 2.8" panels using the ILI9341 controller • SSD1963_4 4.3" panels using the SSD1963controller • SSD1963_5 5" panels using the SSD1963controller • SSD1963_5A alternative version of the 5" panel • SSD1963_7 7" panels using the SSD1963controller • SSD1963_7A alternative version of the 7" panel 'orientation' can be LANDSCAPE, PORTRAIT, RLANDSCAPE or RPORTRAIT. These can be abbreviated to L, P, RL or RP. The R prefix indicates the reverse or "upside down" orientation. **ILI9341, ST7735 and ILI9163 based panels:** 'C/D pin' and 'reset pin' are the Micromite I/O pins to be used for these functions. Any free pin can be used. 'CS pin' can also be any I/O pin but is optional. If a touch controller is not used this parameter can be left off the command and the CS pin on the LCD display wired permanently to ground. If the touch controller is used this pin must then be specified and connected to a Micromite I/O pin. NOTE: The CPU speed must be 20MHz or greater. **For SSD1963 based panels:** 'LCD-A pin' can also be any I/O pin but is optional. This can be used to control the brightness of the backlight if the internal SSD1963 controller is not used. 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 LCD panel is no longer required the command OPTION LCDPANEL DISABLE can be used to disable the LCD panel feature and return the I/O pins for general use.
OPTION LCDPANEL CONSOLE [font [, fc [,bc blight]]] or OPTION LCDPANEL NOCONSOLE	Configures the LCD display panel for use as the console output. The LCD must be one of the SSD1963 versions in the landscape or reverse landscape orientation and it must be first configured using OPTION LCDPANEL SSD1963_x (above). 'font' is the default font, 'fc' is the default foreground colour, 'bc' is the default background colour and 'blight' is the default backlight brightness (2 to 100).

	These parameters are optional and default to font 2, white, black and 100%. These settings are applied at power up. Colour coding in the editor is also turned on by this command (OPTION COLOURCODE OFF will turn it off again). This setting is saved in flash and will be automatically applied on startup. To disable it use the OPTION LCDPANEL NOCONSOLE command.
OPTION RTC Data-pin, Clock-pin or OPTION RTC DISABLE	Configures the Micromite Plus for access to a Real Time Clock (RTC). 'Data-pin' is the I/O pin number used for the I^2C data line and Clock-pin is the same for the I^2C clock. If they are the same as the standard I^2C interface then that interface will be used. If they are different a private I^2C interface will be created to communicate with the RTC (this frees up the main I^2C interface for other tasks). When this command is used the settings will be saved in non volatile memory and automatically applied at startup or reset. Also on startup MMBasic will automatically query the RTC for the current time and date and set the internal clock (similar to the RTC GETTIME command). The general commands for accessing the RTC (RTC SETTIME, RTC GETREG, etc) can still be used and will access the RTC connected to the I/O pins specified in this command. The command OPTION RTC DISABLE can be used to disable this feature and return the I/O pins for general use.
OPTION SDCARD CS-pin [, CD-pin [,WP pin]] or OPTION SDCARD DISABLE	Configures the Micromite Plus to access an attached SD card. 'CS-pin' is the I/O pin number that will be used as chip select (pin 1 on the SD card connector). 'CD-pin' is optional and is the I/O pin number that will be used to connect to the card detect pin on the SD card connector. The Micromite will provide a weak pullup on this pin which is switched to ground when a card is inserted. If this signal is not provided the Micromite Plus will need to be restarted if the SD card was changed. 'WP-pin' is optional and is the I/O pin number that will be used to connect to the write protect pin on the SD card connector. The Micromite will provide a weak pullup on this pin which is switched to ground when a write protected card is inserted. Some SD card connectors reverse the polarity of the 'CD-pin' or 'WP-pin' signal - ie, they are open (and therefore the signal is high) when the card is inserted or write protected. In this case the pin numbers used for 'CD-pin' and 'WP-pin' can be a negative number. This will tell MMBasic to invert the polarity of these signals. This command only needs to be run once. When the Micromite is restarted MMBasic will automatically initialise the SD card interface. If the SD card is no longer required the command OPTION SDCARD DISABLE can be used which will disable the SD card and return the I/O pins for general use.
OPTION TOUCH T_CS pin, T_IRQ pin [, click pin] or OPTION TOUCH DISABLE	Configures the Micromite Plus to suit the touch sensitive feature of an attached LCD panel. 'T_CS pin' and 'T_IRQ pin' are the Micromite I/O pins to be used for chip select and touch interrupt respectively (any free pins can be used). 'click pin' specifies an optional I/O pin that will be driven briefly high when a screen control is touched. This can be used to drive a small Piezo buzzer. This command 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 touch controller. If the touch facility is no longer required the command OPTION TOUCH DISABLE can be used to disable the touch feature and return the I/O pins for general use.

PRINT #fnbr, expression [[,;]expression] … etc	Same as the normal PRINT command except that the output is directed to a file previously opened for OUTPUT or APPEND as '#fnbr'. See the OPEN command.
RMDIR dir$	Remove, or delete, the directory 'dir$' on the SD card.
SAVE [file$]	Saves the program in the current working directory of the SD card as 'file$'. Example: SAVE "TEST.BAS" If an extension is not specified ".BAS" will be added to the file name.
SEEK [#]fnbr, pos	Will position the read/write pointer in a file that has been opened on the SD card for RANDOM access to the 'pos' byte. The first byte in a file is numbered one so `SEEK #5,1` will position the read/write pointer to the start of the file.

Functions (Micromite Plus Only)

CTRLVAL(#ref)	Returns the current value of an advanced control. '#ref' is the control's reference (a number from 1 to 100). For controls like check boxes or switches it will be the number one (true) indicating that the control has been selected by the user or zero (false) if not. For controls that hold a number (eg, a SPINBOX) the value will be the number (normally a floating point number). For controls that hold a string (eg, TEXTBOX) the value will be a string.
CWD$	Returns the current working directory on the SD card as a string.

DIR$(fspec, type)
or
DIR$(fspec)
or
DIR$( )
	Will search an SD card for files and return the names of entries found.
	'fspec' is a file specification using wildcards the same as used by the FILES command. Eg, "*.*" will return all entries, "*.TXT" will return text files.
	'type' is the type of entry to return and can be one of:
	VOL Search for the volume label only
	DIR Search for directories only
	FILE Search for files only (the default if 'type' is not specified)
	The function will return the first entry found. To retrieve subsequent entries use the function with no arguments. ie, DIR$( ). The return of an empty string indicates that there are no more entries to retrieve.
	This example will print all the files in a directory:
	`f$ = DIR$("*.*", FILE)`
	`DO WHILE f$ <> ""`
	`PRINT f$`
	`f$ = DIR$()`
	`LOOP`
	You must change to the required directory before invoking this command.

EOF([#]fnbr)	Will return true if the file previously opened on the SD card for INPUT with the file number '#fnbr' is positioned at the end of the file. The # is optional. Also see the OPEN, INPUT and LINE INPUT commands and the INPUT$ function.
INPUT$(nbr, [#]fnbr)	Will return a string composed of 'nbr' characters read from a file on the SD card previously opened for INPUT with the file number '#fnbr'. This function will read all characters including carriage return and new line without translation. The # is optional. Also see the OPEN command.
LOC([#]fnbr)	For a file on the SD card opened as RANDOM this will return the current position of the read/write pointer in the file. Note that the first byte in a file is numbered 1. The # is optional.
TOUCH(DOWN)	Will return true if the screen is currently being touched.
TOUCH(UP)	Will return true if the screen is currently NOT being touched.
TOUCH(LASTX)	Will return the X coordinate of the last location that was touched.

TOUCH(LASTY)	Will return the Y coordinate of the last location that was touched.
TOUCH(REF)	Will return the reference number of the control that is currently being touched or zero if no control is being touched.
TOUCH(LASTREF)	Will return the reference number of the last control that was touched.