Armmite L4
User
Manual

MMBasic Ver 5.05.09

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

or http://thebackshed.com

The Armmite L4 is a new addition to the Micromite family using the 32, 48 and 64-pin STM32L43x microcontrollers. The Armmite L4 firmware implements most features of the standard Micromite as described in the *Micromite User Manual*. It has a many additional features, some differences and some omissions and they are described in this document.

The focus of this manual is to describe just the features that are **unique** to the Armmite L4. 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 features in the Armmite L4.

The Armmite L4 is a port of the standard Micromite with some features from the Micromite Plus and the Micromite eXtreme. This includes features of the BASIC language, input/output, communications, etc. Some commands have changed slightly (for example the CPU command) but for the main part Micromite programs will run unchanged on the Armmite L4. The following summarises additional features in the Armmite L4 as compared to the standard Micromite:

STM32L43x Processor

The Armmite L4 is based on the STM32L43x series of microcontrollers. These chips are available in 32, 48 and 64-pin surface mount packages and have similar program space and RAM to the standard Micromite.

High Speed Single Precision Floating Point

The Armmite L4 uses the built in hardware floating point capability of the STM32L4 which is much faster than floating point on the standard Micromite and uses single precision floating point.

I/O Pins

The 64-pin Armmite L4 has 47 free I/O pins with 14 analogue capable. The 48-pin Armmite L4 has 33 free I/O pins with 8 analog capable and the 32-pin chip has 21 free I/O pins with 9 analogue capable. All analogue pins use a 12-bit analogue to digital conversion rather than 10-bit on the standard Micromite.

The Armmite L4 has one I^2C ports, three SPI ports (two on the 32 pin device), five PWM channels and two hardware serial COM ports (one on the 32-pin device).

High Speed LCD Panels

The Armmite L4 supports thirteen different sized LCD display panels from 0.91" to 4". The range of drivers for monochrome displays use a memory based framebuffer. This allows complex updates to be written whilst minimising screen flashing and tearing. The Armmite L4 supports enhanced drawing commands including ARCs, TRIANGLEs, and POLYGONs.

Flash Memory File System

The Armmite L4 includes a comprehensive implementation of LittleFS to provide a file capability for the Armmite L4. The file storage medium is an 8-pin SPI flash memory chip. Supported chips are the Winbond W25Q32, the Winbond W25Q64, and the Winbond W25Q128 providing between 4 and 16 Mbytes of storage.

Limitations/differences compared to the full SDcard implementation on the Micromite Plus are:

Single file open at a time

Single top level directory

Filenames are limited to 31 characters

Filenames are case sensitive

Built-in Real time clock (RTC)

The Armmite L4 includes a built-in RTC. This can be separately powered (not 32-pin part). All time and date functions work directly with the RTC and the timing can be trimmed with an OPTION command. The real time clock can be read at millisecond precision. The 32,768Hz crystal for the RTC is also used to discipline the main CPU oscillator ensuring accuracy of commands like TIMER.

Low Power operations

The Armmite L4 supports very low power operation ideal for battery applications

12-Bit DACs and ADCs

The Armmite L4 supports 2 12-bit DACs which can be set individually and programmed to output any arbitrary waveform. All ADCs are 12-bit and in addition ADC pairs can be used for differential measurement.

Comparison to Micromite 5.05.01

Missing commands

RANDOMIZE : not needed as Armmite L4 has H/W random number generator

TRON: deprecated – use TRACE ON

TROFF: deprecated = use TRACE OFF

WHILE:/WEND: deprecated – use DO WHILE/ LOOP

RTC: not needed Armmite L4 has H/W RTC, update is done through TIME$ and DATE$ commands

CFUNCTION/END CFUNCTION: not supported on Armmite L4

CSUB/END CSUB: not supported on Armmite L4

DS18B20: deprecated = use TEMPR

LIBRARY: not supported on Armmite L4

Added Commands

ARC: see command listing for details

DAC: see command listing for details

FILES: Armmite L4 supports a full file system on a flash memory chip: see command listing
FORMAT: Armmite L4 supports a full file system on a flash memory chip : see command listing
HUMID: removed from Micromite in 5.04.01, see command listing for details

IR SEND: removed from Micromite in 5.04.01, see command listing for details

KILL: Armmite L4 supports a full file system on a flash memory : see command listing LOAD: Armmite L4 supports a full file system on a flash memory chip : see command listing LONGSTRING: see command listing for details

NAME: Armmite L4 supports a full file system on a flash memory chip : see command listing
POLYGON: see command listing for details

REFRESH: see command listing for details

SAVE: Armmite L4 supports a full file system on a flash memory chip : see command listing

SEEK: Armmite L4 supports a full file system on a flash memory chip : see command listing SPI2: see command listing for details

SPI3: see command listing for details

SYNCH: see command listing for details

TRIANGLE: see command listing for details

WS2812: see command listing for details

Command Differences

CPU: see section on low power operations for details

DATE$: Sets the RTC

I2C: see command listings for details

OPTION: see command listings for details

PIN: additional functionality: see command listings for details

SETPIN: additional functionality: see command listings for details

TEXT: Supports Micromite Plus formatting including rotated text

TIME$: Sets the RTC

Missing Functions

DS18B20: deprecated = use TEMPR

Added Functions

ATAN2: see function listing for details
BIN2STR$(): see function listing for details
DATETIME$: see function listing for details
DAY$: see function listing for details
DIFF: see function listing for details
DIR$: Armmite L4 supports a full file system on a flash memory , see function listing for details
DISTANCE: removed from Micromite in 5.04.01, see function listing for details
EPOCH: see function listing for details
FIELD: see function listing for details
FILESIZE: see function listing for details
LCOMPARE: see function listing for details
LGETSTR: see function listing for details
LINSTR: see function listing for details
LLEN: see function listing for details
MM.BOOTCOUNT: see function listing for details
MM.FREE: see function listing for details
SPI2: see function listing for details
SPI3: see function listing for details
STR2BIN(): see function listing for details

Function Differences

DATE$: Reads the RTC
PIN: additional functionality, see function listing for details
TIME$: Reads the RTC

Micromite Family Summary

The Micromite Family consists of three major types, the standard Micromite, the Micromite Plus and the Armmite L4. All use the same BASIC interpreter and have the same basic capabilities however they differ in the number of I/O pins, the amount of memory, the displays that they support and their intended use.

Standard Micromite	Comes in a 28-pin or 44-pin package and is designed for small embedded controller applications and supports small LCD display panels. The 28-pin version is particularly easy to use as it is easy to solder and can be plugged into a standard 28-pin IC socket.
Micromite Plus	This uses a 64-pin and 100-pin TQFP surface mount package and supports a wide range of touch sensitive LCD display panels from 1.44" to 8" in addition to the standard features of the Micromite. It is intended as a sophisticated controller with easy to create on-screen controls such as buttons, switches, etc.
Armmite L4	This comes in 32, 48-pin and 64-pin TQFP and TQFN surface mount packages. Overall performance of the Armmite L4 at 80MHz clock speed is greater or equal to the Micromite at 48MHz. The Armmite L4 provides a useful alternative to the Micromite particularly where battery powered applications are intended.

	Micromite		**Micromite Plus**		**Armmite L4**		
	28-pin DIP	44-pin SMD	64-pin SMD	100-pin SMD	64-pin SMD	48-pin SMD	32-pin SMD
Maximum CPU Speed	48 MHz	48 MHz	120 MHz	120 MHz	80MHz	80 MHz	80 MHz
Maximum BASIC Program Size	59 KB	59 KB	100 KB	100 KB	52 KB-308KB *	52 KB	52 KB
RAM Memory Size	52 KB	52 KB	108 KB	108 KB	50 KB-256KB *	50 KB	50 KB
Clock Speed (MHz)	5 to 48	5 to 48	5 to 120	5 to 120	2 to 80	2 to 80	2 to 80
Total Number of I/O pins	19	33	45	77	47	33	21
Number of Analog Inputs	10	13	28	28	14	8	9
Number of Serial I/O ports	2	2	3 or 4	3 or 4	2	2	1
Number of SPI Channels	1	1	2	2	3	3	2
Number of I^2C Channels	1	1	1 + RTC	1 + RTC	1	1	1
Number of 1-Wire I/O pins	19	33	45	77	45	33	19
PWM or Servo Channels	5	5	5	5	5	5	5
Serial Console	✓	✓	✓	✓	✓	✓	✓
USB Console			✓	✓			
PS2 Keyboard and LCD Console			✓	✓			

SD Card Interface			✓	✓	Flash	Flash	Flash
ILI9341 TFT Displays	✓	✓	✓	✓	✓	✓	✓
Supports Ten LCD Panels from 1.44" to 8" (diameter)			✓	✓	13 to 4”	13 to 4”	13 to 4”
Floating Point Precision	Single S/W	Single S/W	Double S/W	Double S/W	Single H/W	Single H/W	Single H/W
Power Requirements	3.3V 30 mA	3.3V 30 mA	3.3V 80 mA	3.3V 80 mA	3.3V 20 mA	3.3V 20 mA	3.3V 20 mA

Suitable Microcontrollers

The microcontroller used in the Armmite L4 is the STM32L43x series manufactured by ST. The firmware will automatically adjust for the 32, 48, or 64 pin versions.

The recommended chips are:

STM32L432KCU6	UFQFPN32 - 32-pin, 5x5 mm, 0.5 mm pitch
STM32L431KCU6	UFQFPN32 - 32-pin, 5x5 mm, 0.5 mm pitch
STM32L431CCU6	UFQFPN48 - 48-lead, 7x7 mm, 0.5 mm pitch
STM32L431CCT6	LQFP48 - 48-pin, 7 x 7 mm, 0.5 mm pitch
STM32L431RCT6	LQFP64 - 64-pin, 10 x 10 mm, 0.5 mm pitch
STM32L433CCU6	UFQFPN48 - 48-lead, 7x7 mm, 0.5 mm pitch
STM32L433CCT6	LQFP48 - 48-pin, 7 x 7 mm, 0.5 mm pitch
STM32L433RCT6	LQFP64 - 64-pin, 10 x 10 mm, 0.5 mm pitch

* The firmware should also run on the following processors
STM32L475RGT6, STM32L475RET6, STM32L476RGT6, STM32L476RET6,
STM32L496RGT6, STM32L496RET6
But limited/no testing has been carried out.

32-pin Test and Development Board

The easiest (and cheapest way to get up and running with the Armmite L4 is to purchase the NUCLEO-L432KC. This is a STM32 Nucleo-32 development board with STM32L432KC MCU and includes a ST-LINK/V2-1 debugger/programmer. The firmware supports referencing pins by using the Arduino designations as well as the pin number. e.g. SETPIN A0,AIN

Armmite L4: MM2 compatible PCB

This little PCB will plug into a standard 28-pin DIP socket exactly like a MM2. The difference is that it has on board an Armmite L4 with in-built RTC and also a 16Mbyte Flash drive for storing programs and data using the Armmite's Flash file manager and, of course, it can run at ultra-low power consumption. Gerbers and details are available online at :
https://www.thebackshed.com/forum/forum_posts.asp?TID=11124

Armmite L4: Backpack PCB

This PCB is fully described at https://www.thebackshed.com/forum/forum_posts.asp?TID=10981
Gerbers for the latest version are available online at :
https://www.thebackshed.com/forum/forum_posts.asp?TID=11130&KW=gerber

Nucleo L476RG

The firmware will run on a Nucleo-L476RG PCB. The STM32L476RG processor gives additional RAM (96Kb usable by MMBasic) and FLASH memory over the STML43x series devices. The firmware supports referencing pins on the Nucleo-L476RG by using the Arduino designations as well as the pin number. e.g. SETPIN A0,AIN. Testing of the firmware on the STM32L476RG has been limited as at 22/3/2019.

Minimum Circuits

32-pin chip

Connect GND to pins 16 and 32

Connect 3.3V to pins 1, 5, and 17

Tie BOOT0 (pin 31) to GND with a 10K resistor

Connect a low capacitance (12pf maximum) 32768Hz crystal between pins 2 and 3. Decouple both pins to GND with 12pf capacitors

Tie NRST (pin 4) to 3.3V with a 10K resistor

Decouple the 3.3V supply with 0.1uF capacitors close to pins 1 & 32 and 16 & 17

Connect a USB/UART TX pin to STM32L4 pin 25

Connect a USB/UART RX pin to STM32L4 pin 8

Connect a USB/UART GND pin to GND

48-pin chip

Connect GND to pins 8, 23, 35, and 47

Connect 3.3V to pins 9, 24, 36, and 48

Tie BOOT0 (pin 44) to GND with a 10K resistor

Connect a low capacitance (12pf maximum) 32768Hz crystal between pins 3 and 4. Decouple both pins to GND with 12pf capacitors

Tie NRST (pin 7) to 3.3V with a 10K resistor

Decouple the 3.3V supply with 0.1uF capacitors close to pins 8 & 9, 23 & 24, 35 & 36, 47 & 48

Connect a USB/UART TX pin to STM32L4 pin 13

Connect a USB/UART RX pin to STM32L4 pin 12

Connect a USB/UART GND pin to GND

Connect VBAT (pin 1) to 3.3V or to a 3V battery which has a common ground with the rest of the circuit. Using a battery will keep the RTC accurate even if the rest of the circuit is powered off

64-pin chip

Connect GND to pins 12, 31, 47, and 63

Connect 3.3V to pins 13, 32, 48, and 64

Tie BOOT0 (pin 60) to GND with a 10K resistor

Connect a low capacitance (12pf maximum) 32768Hz crystal between pins 3 and 4. Decouple both pins to GND with 12pf capacitors

Tie NRST (pin 7) to 3.3V with a 10K resistor

Decouple the 3.3V supply with 0.1uF capacitors close to pins 8 & 9, 23 & 24, 35 & 36, 47 & 48

Connect a USB/UART TX pin to STM32L4 pin 17

Connect a USB/UART RX pin to STM32L4 pin 16

Connect a USB/UART GND pin to GND

Connect VBAT (pin 1) to 3.3V or to a 3V battery which has a common ground with the rest of the circuit. Using a battery will keep the RTC accurate even if the rest of the circuit is powered off

Programming the Firmware

Programming the 32, 48 and 64-pin Armmite L4 with a ST-LINK programmer

Download the free ST-LINK software and wire the processor as follows:

ST-LINK Pins	Description	64-pin Armmite L4 pin numbers	48-pin Armmite L4 pin numbers	32-pin Armmite L4 pin numbers
1 - VCC	Power Supply (3.3V)	1, 13, 32, 48, 64	1, 9, 24, 36, 48	1, 5, 17
2 - SWCLK	Programming Clock	49	37	24
3 - GND	Ground	12, 31, 47, 63	8, 23, 35, 47	16, 32
4 - SWDIO	Programming Data	46	34	23
5 - NRST	Master Reset (active low)	7	7	4
6 - NC	Not used			

Notes:

- A pullup resistor of 10K is required between NRST and VCC.

Programming the Nucleo-L432KC

Copy the supplied .bin file to the disk NODE-L432KC on a PC

Programming over a UART

All STM32L4 chips have an in-built bootloader and can be programmed using the free STM32CubeProgrammer software. To activate this, the BOOT0 pin is tied to VCC rather than GND.

In the case of the 48-pin and 64-pin STM32L4 chips UART2 is the Micromite Console so without any wiring changes, other than to tie BOOT0 high, it is possible to program the chip with new firmware simply by running STM32CubeProgrammer instead of your usual terminal emulator.

Unfortunately the NUCLEO-L432KC uses non-standard pins for the console but you can program this by connecting to the Armmite COM1 connections. For the 48 and 64-pin parts you would normally use the console pins, although com1 or com2 pins will also work.

64-pin Armmite L4 Pinouts

Pin	Features					Port
1	VBAT					
2	DIGITAL_IN	DIGITAL_OUT	USER-BUTTON*			C13
3	OSC32					C14
4	OSC32					C15
5	DIGITAL_IN	DIGITAL_OUT				H0
6	DIGITAL_IN	DIGITAL_OUT				H1
7	NRST					
8	ANALOG_IN_1	DIGITAL_IN	DIGITAL_OUT			C0
9	ANALOG_IN_2	DIGITAL_IN	DIGITAL_OUT			C1
10	ANALOG_IN_3	DIGITAL_IN	DIGITAL_OUT			C2
11	ANALOG_IN_4	DIGITAL_IN	DIGITAL_OUT			C3
12	VSSA					
13	VDDA					
14	ANALOG_IN_5	DIGITAL_IN	DIGITAL_OUT	COUNT	IR	A0
15	ANALOG_IN_6	DIGITAL_IN	DIGITAL_OUT	SPI_CLK		A1
16	CONSOLE_TX					A2
17	CONSOLE_RX					A3
18	VSS					
19	VDD					
20	ANALOG_IN_9	DIGITAL_IN	DIGITAL_OUT	DAC1	GREEN-LED*	A4
21	ANALOG_IN_10	DIGITAL_IN	DIGITAL_OUT	DAC2		A5
22	ANALOG_IN_11	DIGITAL_IN	DIGITAL_OUT	SPI_IN		A6
23	ANALOG_IN_12	DIGITAL_IN	DIGITAL_OUT	SPI_OUT		A7
24	ANALOG_IN_13	DIGITAL_IN	DIGITAL_OUT			C4
25	ANALOG_IN_14	DIGITAL_IN	DIGITAL_OUT			C5
26	ANALOG_IN_15	DIGITAL_IN	DIGITAL_OUT			B0
27	ANALOG_IN_16	DIGITAL_IN	DIGITAL_OUT	COUNT		B1
28	DIGITAL_IN	DIGITAL_OUT				B2
29	DIGITAL_IN	DIGITAL_OUT	COM2_TX			B10
30	DIGITAL_IN	DIGITAL_OUT	COM2_RX			B11
31	VSS					
32	VDD					
33	DIGITAL_IN	DIGITAL_OUT				B12
34	DIGITAL_IN	DIGITAL_OUT	SPI3_CLK			B13
35	DIGITAL_IN	DIGITAL_OUT	SPI3_IN			B14
36	DIGITAL_IN	DIGITAL_OUT	SPI3_OUT			B15
37	DIGITAL_IN	DIGITAL_OUT				C6

38	DIGITAL_IN	DIGITAL_OUT			C7
39	DIGITAL_IN	DIGITAL_OUT			C8
40	DIGITAL_IN	DIGITAL_OUT			C9
41	DIGITAL_IN	DIGITAL_OUT	PWM_1A		A8
42	DIGITAL_IN	DIGITAL_OUT	COM1_TX	PWM_1C	A9
43	DIGITAL_IN	DIGITAL_OUT	COM1_RX	PWM_1D	A10
44	DIGITAL_IN	DIGITAL_OUT	PWM_1B		A11
45	DIGITAL_IN	DIGITAL_OUT	COM1_DE	COUNT	A12
46	DIGITAL_IN	DIGITAL_OUT	SWDIO		A13
47	VSS				
48	VDD				
49	DIGITAL_IN	DIGITAL_OUT	SWCLK		A14
50	DIGITAL_IN	DIGITAL_OUT			A15
51	DIGITAL_IN	DIGITAL_OUT			C10
52	DIGITAL_IN	DIGITAL_OUT			C11
53	DIGITAL_IN	DIGITAL_OUT			C12
54	DIGITAL_IN	DIGITAL_OUT			D2
55	DIGITAL_IN	DIGITAL_OUT	SPI2_CLK		B3
56	DIGITAL_IN	DIGITAL_OUT	SPI2_IN		B4
57	DIGITAL_IN	DIGITAL_OUT	SPI2_OUT		B5
58	DIGITAL_IN	DIGITAL_OUT	I2C_SCL		B6
59	DIGITAL_IN	DIGITAL_OUT	I2C_SDA		B7
60	BOOT0				H3
61	DIGITAL_IN	DIGITAL_OUT	PWM_2A		B8
62	DIGITAL_IN	DIGITAL_OUT			B9
63	VSS				
64	VDD				

* Nucleo-L476RG

48-pin Armmite L4 Pinouts

Pin	Features					Port
1	VBAT					
2	DIGITAL_IN	DIGITAL_OUT				C13
3	OSC32					C14
4	OSC32					C15
5	DIGITAL_IN	DIGITAL_OUT				H0
6	DIGITAL_IN	DIGITAL_OUT				H1
7	NRST					
8	VSSA					
9	VDDA					
10	ANALOG_IN_5	DIGITAL_IN	DIGITAL_OUT	COUNT	IR	A0
11	ANALOG_IN_6	DIGITAL_IN	DIGITAL_OUT	SPI_CLK		A1
12	CONSOLE_TX					A2
13	CONSOLE_RX					A3
14	ANALOG_IN_9	DIGITAL_IN	DIGITAL_OUT	DAC1		A4
15	ANALOG_IN_10	DIGITAL_IN	DIGITAL_OUT	DAC2		A5
16	ANALOG_IN_11	DIGITAL_IN	DIGITAL_OUT	SPI_IN		A6
17	ANALOG_IN_12	DIGITAL_IN	DIGITAL_OUT	SPI_OUT		A7
18	ANALOG_IN_15	DIGITAL_IN	DIGITAL_OUT			B0
19	ANALOG_IN_16	DIGITAL_IN	DIGITAL_OUT	COUNT		B1
20	DIGITAL_IN	DIGITAL_OUT				B2
21	DIGITAL_IN	DIGITAL_OUT	COM2_TX			B10
22	DIGITAL_IN	DIGITAL_OUT	COM2_RX			B11
23	VSS					
24	VDD					
25	DIGITAL_IN	DIGITAL_OUT				B12
26	DIGITAL_IN	DIGITAL_OUT	SPI3_CLK			B13
27	DIGITAL_IN	DIGITAL_OUT	SPI3_IN			B14
28	DIGITAL_IN	DIGITAL_OUT	SPI3_OUT			B15
29	DIGITAL_IN	DIGITAL_OUT	PWM_1A			A8
30	DIGITAL_IN	DIGITAL_OUT	COM1_TX	PWM_1C		A9
31	DIGITAL_IN	DIGITAL_OUT	COM1_RX	PWM_1D		A10
32	DIGITAL_IN	DIGITAL_OUT	PWM_1B			A11
33	DIGITAL_IN	DIGITAL_OUT	COM1_DE	COUNT		A12
34	DIGITAL_IN	DIGITAL_OUT	SWDIO			A13
35	VSS					
36	VDD					
37	DIGITAL_IN	DIGITAL_OUT	SWCLK			A14

38	DIGITAL_IN	DIGITAL_OUT		A15
39	DIGITAL_IN	DIGITAL_OUT	SPI2_CLK	B3
40	DIGITAL_IN	DIGITAL_OUT	SPI2_IN	B4
41	DIGITAL_IN	DIGITAL_OUT	SPI2_OUT	B5
42	DIGITAL_IN	DIGITAL_OUT	I2C_SCL	B6
43	DIGITAL_IN	DIGITAL_OUT	I2C_SDA	B7
44	BOOT0			H3
45	DIGITAL_IN	DIGITAL_OUT	PWM_2A	B8
46	DIGITAL_IN	DIGITAL_OUT		B9
47	VSS			
48	VDD			

32-pin Armmite L4 Pinouts

Pin	Features					Port
1	VDD					
2	OSC32					C14
3	OSC32					C15
4	NRST					
5	VDD					
6	ANALOG_IN_5	DIGITAL_IN	DIGITAL_OUT	COUNT	IR	A0
7	ANALOG_IN_6	DIGITAL_IN	DIGITAL_OUT	SPI_CLK		A1
8	CONSOLE_TX					A2
9	ANALOG_IN_8	DIGITAL_IN	DIGITAL_OUT	PWM_2A		A3
10	ANALOG_IN_9	DIGITAL_IN	DIGITAL_OUT	DAC1		A4
11	ANALOG_IN_10	DIGITAL_IN	DIGITAL_OUT	DAC2		A5
12	ANALOG_IN_11	DIGITAL_IN	DIGITAL_OUT	SPI_IN		A6
13	ANALOG_IN_12	DIGITAL_IN	DIGITAL_OUT	SPI_OUT		A7
14	ANALOG_IN_15	DIGITAL_IN	DIGITAL_OUT			B0
15	ANALOG_IN_16	DIGITAL_IN	DIGITAL_OUT	COUNT		B1
16	VSS					
17	VDD					
18	DIGITAL_IN	DIGITAL_OUT	PWM_1A			A8
19	DIGITAL_IN	DIGITAL_OUT	COM1_TX	PWM_1C		A9
20	DIGITAL_IN	DIGITAL_OUT	COM1_RX	PWM_1D		A10
21	DIGITAL_IN	DIGITAL_OUT	PWM_1B			A11
22	DIGITAL_IN	DIGITAL_OUT	COM1_DE	COUNT		A12
23	DIGITAL_IN	DIGITAL_OUT	SWDIO			A13
24	DIGITAL_IN	DIGITAL_OUT	SWCLK			A14
25	CONSOLE_RX					A15
26	DIGITAL_IN	DIGITAL_OUT	SPI2_CLK	GREEN_LED*		B3
27	DIGITAL_IN	DIGITAL_OUT	SPI2_IN			B4
28	DIGITAL_IN	DIGITAL_OUT	SPI2_OUT			B5
29	DIGITAL_IN	DIGITAL_OUT	I2C_SCL			B6
30	DIGITAL_IN	DIGITAL_OUT	I2C_SDA			B7
31	BOOT0					H3
32	VSS					

* On Nucleo-L432KC PCB

Low Power Operations

One unique characteristic of the STM32L4 range is the low power consumption of the microprocessor making it ideal for battery based applications as well as those with supplied power.

Current draw

The Table below compares the PIC32MX170 with the Armmite L4 at roughly equivalent performance levels. The Armmite values are actual measurements made on the Armmite L4: Backpack PCB pictured above:

Micromite CPU speed	PIC Current Draw	Armmite L4 CPU speed	Armmite Current
48MHz	31mA	80MHz	12.7mA
40MHz	26mA		
30MHz	21mA	48MHz	7.5MHz
20Mhz	15mA	32MHz	5.7mA
10MHz	10mA	16MHz	2.8mA
5MHz	6mA	8MHz	1.7mA
		2MHz	0.8mA
PAUSE	As per CPU speed	PAUSE	1.1mA
CPU SLEEP	40uA	CPU SLEEP	3uA*

* Under optimum conditions when no additional peripherals are started.

CPU speed settings

The Armmite L4 supports the following CPU speeds:

2, 4, 8, 16, 24, 32, 48, 64

with the normal Micromite command “CPU speed” e.g. CPU 80

The STM32L4 has multiple internal clocks and the UARTs and I2C are driven off a 16MHz clock separate from the main processor. This ensures that changing CPU speed will have no impact on I2C or UART operations.

SPI and PWM are locked to the main processor clock so will change with the CPU speed.

The Armmite L4 firmware will automatically change the CPU speed transparently to the user if devices like DS18B20 or WS2812 are used if the current CPU speed is inadequate for the device to operate properly. The speed change is done for the minimum time possible so, for example, there will be a very brief change when TEMPR START is issued and another when the TEMPR() function is called

The firmware will also slow the clock during PAUSE commands unless a COM port is open or a PWM channel is running. This has a big impact on current draw as shown in the table above.

CPU SLEEP command

The CPU sleep command operates as per the Micromite with the following exceptions:

CPU SLEEP.

The wakeup pin is as per the pinout above. However any other COUNT pin can also be used to wake the processor if it is enabled with SETPIN pinno, CIN or PIN or FIN.

CPU SLEEP time.

The Armmite uses the RTC to generate an interrupt to wake the processor after a period of sleep. Any period can be specified including fractions of seconds and because the RTC is used the timing will be accurate. Using the embedded ARMMITE L4 date and time functions makes it easy to sleep until any particular time. e.g.

Midnight_tonight% = epoch(date$+" 00:00:00")+86400 'epoch at start of day today + secs in a day

CPU SLEEP Midnight_tonight% - epoch(now) ' sleep until midnight tonight

Managing Current Draw

In order to minimise current draw peripheral clocks for devices like I2C are not started unless the device is used. Each additional device started will add minimally to the current draw so it is important to minimise device usage and turn them off when not in use.

Obviously if you use a pin to drive an LED at 10mA then current draw will be 10mA greater than the minimum processor usage but some other impacts and effects of external peripherals are less obvious.

- External voltage regulators can have significant quiescent currents. For example the ubiquitous LT1117 has a typical quiescent current of 5mA so even if the processor is set to sleep the circuit will still draw 5mA
- An IR receiver takes 700uA
- OLED displays will draw current depending on the number of pixels lit
- The "old" Nokia 5110 display seems to be very efficient. I have demonstrated the Armmite L4 CPU in SLEEP mode with a Nokia 5110 displaying time, date and temperature at 290uA
- Measuring current draw is difficult at low values. You can't measure supply power with anything else external and separately powered connected: Power usage in the example above went down from 290uA to 250uA when the USB/UART was connected, presumably through parasitic supply through the Console TX pin

Flash File System Support

The Armmite L4 has support for a complete file system using an 8-pin SPI flash memory chip. This includes opening files for reading, writing or random access and loading and saving programs. Supported chips are the Winbond W25Q32, the Winbond W25Q64, and the Winbond W25Q128 providing between 4 and 16 Mbytes of storage.

The connections required to the SPI chip are as follows:

Pin 1 : Chip select, connect to any free I/O pin on the Armmite L4
Pin 2 : SPI data out, connect to SPI2-IN (D12) on the Armmite L4
Pin 3 : _Write enable, connect to GND
pin 4 : GND, connect to GND
pin 5 : SPI data in, connect to SPI2-OUT (D11) on the Armmite L4
pin 6 : SPI clock, connect to SPI2-CLK (D13) on the Armmite L4
pin 7 : RESET, connect to NRST on the Armmite L4
pin 8 : VCC, connect to 3.3V

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.

The system is enabled using the OPTION command “OPTION FLASH cs” where cs is the pin number on the Armmite L4 connected to the Chip select pin of the flash memory chip.

The Armmite L4 flash file system works the same as the SDcard on the Micromite Plus and users should refer to the Micromite Plus Manual for details with the following exceptions:

Limitations/differences compared to the Micromite Plus implementation are:

- Single file open at a time
- Single top level directory
- Filenames are limited to 31 characters
- Filenames are case sensitive

Display support

The Armmite L4 supports a wide range of displays.

SSD1306 controller 128x64 pixel OLED, 0.96” and 1.3” with I2C I/F, monochrome

SSD1306 controller 128x32 pixel OLED, 0.1” with I2C I/F, monochrome

SSD1306 controller 128x64 pixel OLED, 0.96” and 1.3” with SPI I/F, monochrome

SSD1306 controller 128x32 pixel OLED, 0.1” with SPI I/F, monochrome

ST7920 controller 128x64 pixel LCD, 3.25” with SPI I/F, monochrome

Nokia 5110 controller 84x48 pixel LCD, with SPI I/F, monochrome

GDEH029A1 controller 128x296 pixel e-ink, with SPI I/F, monochrome

ILI9163 controller 128x128 pixel TFT, 1.44” with SPI I/F, RGB565

ILI9341 controller 320x240 pixel TFT, 2.2”, 2.4”, 2.8” with SPI I/F, RGB565

ST7735 controller 160x128 pixel TFT, 1.8” with SPI I/F, RGB565

ST7735S controller 160x80 pixel IPS, 0.96” with SPI I/F, RGB565

SSD1331 controller 96x64 pixel OLED, 0.95” with SPI I/F, RGB565

ST7789 controller 240x240 pixel IPS, 1.3” with SPI I/F, RGB565

ILI9481 controller 480x320 pixel TFT, 3.5”, 4” with SPI I/F, RGB565

Connecting SPI Based LCD Panels

The SPI based display controllers share the second SPI channel (SPI2) interface on the Armmite L4 with the touch controller (if present) and the Flash memory 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 or third SPI channel (not STM32L432KC) instead.
The speed of drawing to SPI based displays will be affected by the CPU speed. If this is an issue just bracket the drawing commands with CPU speed changes. e.g. CPU 80: CLS: CPU 4

Connecting I2C Based LCD Panels

The I2C based display controllers use the standard I2C pins as per the pinout for the specific device. Other I2C devices can share the bus subject to their addresses being unique. If an I2C display is configured it will not be necessary to “open” the I2C port for an additional device (I2C OPEN), I2C CLOSE is blocked, and all I2C devices must be capable of 400KHz operation. The I2C bus speed is not affected by changes to the CPU clock speed.

Drawing command support

The Armmite L4 supports all the drawing commands available on the Micromite II. In addition commands are available to draw arcs, polygons, and triangles.

Display Configuration

The details for initialising the various displays are given in the command summary section of this manual – OPTION LCDPANEL displayname, orientation [,display specific parameters]

Monochrome displays

Monochrome displays can have two "colours" zero and not-zero. All displays default such that zero is the natural colour of the display. In the case an OLED this is not-lit. (i.e. black) and non-zero will be the lit colour (normally white, blue, or yellow for OLEDs). In the case of a Nokia 5110 display, zero is the backlight colour and non-zero is black.

All monochrome displays can be used inverted by clearing the display to non-zero e.g. "CLS 1" and then setting 0 as the foreground colour in any graphics commands. The only issue with this is that displays typically have non-functional borders which will remain the native "zero" colour so graphics or text that touches the outside of the active area will merge into the border. TEXT can be output inverted by using zero as the foreground colour and any non-zero value as the background.

All monochrome displays have a memory based frame-buffer which is created when the display is initialised. This will have a small impact on the amount of available RAM for program use (RAM used is width-in-pixels * height-in-pixels / 8 + 256 bytes). Using a frame-buffer gives two significant advantages:

1. Monochrome displays can be powered down to save power and then when power is returned the command “GUI RESET LCDPANEL” will not only re-initialise the display hardware but also restore the display contents

   Updates to the display can be suspended by issuing the command “OPTION AUTOREFRESH OFF”. In this mode drawing and text commands update the framebuffer but not the display. Issuing the command “REFRESH” will write the contents of the framebuffer to the display. This allows complex updates to be made without obvious screen writes being visible. The area written is the smallest rectangle containing all the changes since the last “REFRESH” command”.

Commands (Armmite L4 additions and differences)

ARC x, y, r1, [r2], rad1, rad2, colour	Draws an arc of a circle or a given colour and width between two radials (defined in degrees). Parameters for the ARC command are: x: X coordinate of centre of arc y: Y coordinate of centre of arc r1: inner radius of arc r2: outer radius of arc - can be omitted if 1 pixel wide rad1: start radial of arc in degrees rad2: end radial of arc in degrees colour: Colour of arc
CPU	See the section on Low Power Operations
DAC n, voltage	Sets the DAC channel (1 or 2) to the voltage requested. This command cannot be used if the DACs are in use for waveform output.
DAC START frequency, DAC1array%() [,DAC2array%()]	Sets up the DAC to create an arbitrary waveform. DAC1array%() and optional DAC2array%() should contain numbers in the range 0-4095 to suit the 12-bit DACs. Once started the output continues in the background and control returns to MMBasic. The software automatically and separately uses the number of items in each of the arrays to drive the DACs. The frequency is the rate at which the DACs change value. The maximum frequency is 700KHz. As an example if there are 180 items in the array c%() which are displayed at a frequency of 100,000 Hz this will give a waveform frequency of 100,000/180 = 555Hz. If there are 90 items in the array d%() at the same frequency of 100,000 Hz this will at the same time produce a waveform frequency of 100,000/90 = 1111Hz.
DAC CLOSE	Stops DAC output and returns the DACs to normal use.
DATE$	DATE$ sets the date in the RTC which is used for all subsequent DATE$ function calls

FILES [fspec$]	Lists the files on the Flash memory chip. 'fspec$' (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
FORMAT [pinno]	This command initialises the file system on a FLASH chip connected to SPI2. Supported chips are the Winbond W25Q32, the Winbond W25Q64, and the Winbond W25Q128 providing between 4 and 16 Mbytes of storage. The chip is automatically formatted the first time it is connected so this command would only be used to re-initialise the chip and delete all existing files. If the flash memory is enabled but becomes corrupted it is possible the firmware will not run. In this case reflash the MMBasic firmware and **without** setting the OPTION FLASH command execute FORMAT with the flash CS pin specified. This will re-initialise the flash and allow it to be configured normally again.
HUMID pin, tvar, hvar	Returns the temperature and humidity using the DHT22 sensor. Alternative versions of the DHT22 are the AM2303 or the RHT03 (all are compatible). 'pin' is the I/O pin connected to the sensor. Any I/O pin may be used. 'tvar' is the variable that will hold the measured temperature and 'hvar' is the same for humidity. Both must be present and both must be floating point variables. For example: HUMID 2, TEMP!, HUMIDITY! Temperature is measured in ºC and the humidity is percent relative humidity. Both will be measured with a resolution of 0.1. If an error occurs (sensor not connected or corrupt signal) both values will be 1000.0. The measurement will take 6mS to complete. Normally the signal pin of the DHT22 should be pulled up by a 1K to 10K resistor (4.7K recommended) to the supply voltage. The Armmite will also enable an internal high value pullup resistor so when the cable length is short (under 30cm) the external pullup resistor may be omitted.
I2C CHECK addr	Runs a probe on address “addr”. MM.I2C is set to 0 if the address responds and 1 if it doen’t
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 in the Micromite Manual for details.

IR SEND pin, dev, key	Generate a 12-bit Sony Remote Control protocol infrared signal. 'pin' is the I/O pin to use. This can be any I/O pin which will be automatically configured as an output and should be connected to an infrared LED. Idle is low with high levels indicating when the LED should be turned on. 'dev' is the device being controlled and is a number from 0 to 31, 'key' is the simulated key press and is a number from 0 to 127.The IR signal is modulated at about 38kHz and sending the signal takes about 25mS. The CPU speed must be 10MHz or above
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 in the Micromite Manual for details.
LOAD file$ [,R]	Loads a program called ‘file$’ from the flash file system into program memory. If an extension is not specified “.BAS” will be added to the file name. If the optional parameter “R” or “r” is specified the program will be immediately executed

LONGSTRING APPEND array%(), string$	Append a normal MMBasic string to a long string variable. array%() is a long string variable while string$ is a normal MMBasic string expression.
LONGSTRING CLEAR array%()	Will clear the long string variable array%(). ie, it will be set to an empty string.
LONGSTRING COPY dest%(), src%()	Copy one long string to another. dest%() is the destination variable and src%() is the source variable. Whatever was in dest%() will be overwritten.
LONGSTRING CONCAT dest%(), src%()	Concatenate one long string to another. dest%() is the destination variable and src%() is the source variable. src%() will the added to the end of dest%() (the destination will not be overwritten).
LONGSTRING LCASE array%()	Will convert any uppercase characters in array%() to lowercase. array%() must be long string variable.
LONGSTRING LEFT dest%(), src%(), nbr	Will copy the left hand 'nbr' characters from src%() to dest%() overwriting whatever was in dest%(). ie, copy from the beginning of src%(). src%() and dest%() must be long string variables. 'nbr' must be an integer constant or expression.
LONGSTRING LOAD array%(), nbr, string$	Will copy 'nbr' characters from string$ to the long string variable array%() overwriting whatever was in array%().
LONGSTRING MID dest%(), src%(), start, nbr	Will copy 'nbr' characters from src%() to dest%() starting at character position 'start' overwriting whatever was in dest%(). ie, copy from the middle of src%(). 'nbr' is optional and if omitted the characters from 'start' to the end of the string will be copied src%() and dest%() must be long string variables. 'start' and 'nbr' must be an integer constants or expressions.
LONGSTRING PRINT [#n,] src%()	Prints the longstring stored in src%()
LONGSTRING REPLACE array%() , string$, start	Will substitute characters in the normal MMBasic string string$ into an existing long string array%() starting at position ‘start’ in the long string.
LONGSTRING RIGHT dest%(), src%(), nbr	Will copy the right hand 'nbr' characters from src%() to dest%() overwriting whatever was in dest%(). ie, copy from the end of src%(). src%() and dest%() must be long string variables. 'nbr' must be an integer constant or expression.
LONGSTRING TRIM array%(), nbr	Will trim ‘nbr’ characters from the left of a long string. array%() must be a long string variables. 'nbr' must be an integer constant or expression.
LONGSTRING UCASE array%()	Will convert any lowercase characters in array%() to uppercase. array%() must be long string variable.

NAME old$ AS new$	Rename a file or a directory from ‘old$’ to ‘new$’. Both are strings.
OPEN fname$ FOR mode AS [#]fnbr	Opens a file for reading or writing. ‘fname’ is the filename with an optional extension separated by a dot (.). Long file names with upper and lower case characters are supported. ‘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. Only one file 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.
OPTION AUTOREFRESH mode	When using buffered driver TFT drivers this command controls when screen updates take place. When autorefresh is “ON” updates take place immediately. When autorefresh is “OFF” updates take place in the framebuffer and are only written to the screen when command REFRESH is used.
OPTION FLASH pinno	Defines the pin to be used for the chip select of a flash memory chip to be used for a file system
OPTION LCDPANEL GDEH029A1, orientation, DCpin, RESETpin, CSpin, BUSYpin [,refreshcount]	Initialises an e-Ink display using the GDEH029A1controller. This supports 128 * 296 resolution. See “ OPTION LCDPANEL ILI9341” for details of main parameter usage. ‘BUSYpin’ is an input used by the driver to establish command completion on the slow e-Ink display. An additional parameter refreshcount may be specified to control when the driver does a full update to the display (black/white flash). Default is 1. i.e. the display does a full refresh each write. Setting a bigger value can make updates faster and less obvious but with the risk of permanently creating ghost images on the display.
OPTION LCDPANEL ILI9163, orientation, DCpin, RESETpin, CSpin	Initialises a TFT display using the ILI9163 controller. This supports 128 * 128 resolution. See “ OPTION LCDPANEL ILI9341” for details of parameter usage.

OPTION LCDPANEL ILI9341, orientation, DCpin, RESETpin, CSpin	Configures the Micromite and Micromite Plus to work with an attached TFT panel using the ILI9341 controller. This supports 320 * 240 resolution. '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', 'reset pin' and 'CS pin' are the Micromite I/O pins to be used for these functions. Any free pin can be used.
OPTION LCDPANEL ILI9841, orientation, DCpin, RESETpin, CSpin	Initialises a TFT display using the ILI9841 controller. This supports 480 * 320 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter usage.
OPTION LCDPANEL N5110, orientation, DCpin, RESETpin, CSpin [,contrast]	Initialises a LCD display using the Nokia 5110 controller. This supports 84 * 48 resolution. See " OPTION LCDPANEL ILI9341" for details of main parameter usage. An additional parameter LCDVOP may be specified to control the contrast of the display. Try contrast values between &HA8 and &HD0 to suit your display, default if omitted is &HB1
OPTION LCDPANEL SSD1306I2C, orientation [,offset]	Initialises a OLED display using the SSD1306 controller with an I2C interface. This supports 128 * 64 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter orientation. An additional parameter offset may be specified to control the position of the display. 0.96" displays typically need a value of 0. 1.3" displays typically need a value of 2. Default if omitted is 0. **NB many cheap I2C versions of SSD1306 displays do not implement I2C properly due to a wiring error. This seems to be particularly the case with 1.3" variants**
OPTION LCDPANEL SSD1306I2C32, orientation	Initialises a OLED display using the SSD1306 controller with an I2C interface. This supports 128 * 32 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter orientation.
OPTION LCDPANEL SSD1306SPI, orientation, DCpin, RESETpin, CSpin [,offset]	Initialises a OLED display using the SSD1306 controller with an SPI interface. This supports 128 * 64 resolution. See " OPTION LCDPANEL ILI9341" for details of main parameter usage. An additional parameter offset may be specified to control the position of the display. 0.96" displays typically need a value of 0. 1.3" displays typically need a value of 2. Default if omitted is 0.
OPTION LCDPANEL SSD1331, orientation, DCpin, RESETpin, CSpin	Initialises a colour OLED display using the SSD1331 controller. This supports 96 * 64 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter usage.
OPTION LCDPANEL ST7735, orientation, DCpin, RESETpin, CSpin	Initialises a TFT display using the ST7735 controller. This supports 160 * 128 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter usage.

OPTION LCDPANEL ST7735S, orientation, DCpin, RESETpin, CSpin	Initialises a IPS display using the ST7735S controller. This supports 160 * 80 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter usage.
OPTION LCDPANEL ST7789, orientation, DCpin, RESETpin, CSpin	Initialises a IPS display using the 7789 controller. This supports 240 * 240 resolution. See " OPTION LCDPANEL ILI9341" for details of parameter usage.
OPTION LCDPANEL ST7920, orientation, DCpin, RESETpin	Initialises a LCD display using the ST7920 controller. This supports 128 * 64 resolution. Note this display does not support a chip select so cannot so the SPI bus cannot be shared if this display is used. See " OPTION LCDPANEL ILI9341" for details of other parameter usage.
OPTION MILLESECONDS OFF OPTION MILLISECONDS ON	Default. The time$ function returns the time as "HH:MM:SS" The time$ function returns the time as "HH:MM:SS.MMM"
OPTION RTC CALIBRATE n	Used to calibrate the RTC. n specifies the number of extra clock pulses to be "created" or "removed" every 2^20 real clock pulses. n can be between -511 and + 512. A change of +/-1 should equate to about 0.0824 seconds per day.
OPTION SERIAL PULLUP DISABLE OPTION SERIAL PULLUP ENABLE	permanently stored option that disables pullups on all serial ports default: permanently stored option that enables pullups on all serial ports
OPTION VCC voltage	This allows the user to precisely set the supply voltage to the chip and is used in the calculation of voltages when using analogue inputs. The parameter is not saved and should be initialised either on the command line or in a program. The command OPTION VCC PIN(VDDA) can be used to set VCC based on the internally calibrated reference voltage
PIN(countpinno) = count PIN(dacpinno) = voltage	Sets a pin configured as a counting input (SETPIN n, CIN) to a specific count. This is most useful for zeroing a count when required. Sets a pin configured as a DAC to the voltage specified. The voltage can be between 0 and the value of OPTION VCC (default 3.3V)

POLYGON xarray%(), yarray%() [, bordercolour] [, fillcolour]	Draws a outline or filled polygon defined by the x,y coordinate pairs in xarray%() and yarray%(). If fill colour is omitted then just the polygon outline is drawn. If bordercolour is omitted then it will default to the current gui foreground colour. The polygon should be closed with the first and last elements the same. The size of the arrays should be the same and exactly match the number of x,y coordinate pairs.
PWM 1, freq, 1A [,1B] [,1C] [,1D]	See description of the PWM command in the Micromite User Manual. This command allows the specification of a frequency for up to four channels
PWM 2, freq, 2A	See description of the PWM command in the Micromite User Manual. This command allows the specification of a frequency for one channel
REFRESH	When using buffered display drivers (all monochrome displays) this command forces a screen update
SAVE file$	Saves the program to the flash memory chip as 'file$'. Example: SAVE "TEST.BAS" If an extension is not specified ".BAS" will be added to the file name.
SEEK [#]fnbr, pos	Will position the read/write pointer in a file that has been opened on the Flash memory chip for RANDOM access to the 'pos' byte. The first byte in a file is numbered one so SEEK #5,1 will position the read/write pointer to the start of the file.
SERVO 1 [, freq], 1A [,1B] [,1C] [,1D]	See description of the SERVO command in the Micromite User Manual. This command allows the specification of a frequency for up to four SERVO channels
SERVO 2 [, freq], 2A	See description of the SERVO command in the Micromite User Manual. This command allows the specification of a frequency for a one SERVO channel
SETPIN pinno, DAC	Sets pin 'pinno' as a DAC output. See the pinout listing for the relevant pin numbers for each processor
SPI2 OPEN speed, mode, bits SPI2 READ nbr, array() SPI2 WRITE nbr, data1, data2, data3, … etc or SPI2 WRITE nbr, string$ or SPI2 WRITE nbr, array() SPI2 CLOSE	See Appendix D of the Micromite User Manual. These commands are not available if flash memory and/or the SPI display is configured (with or without TOUCH) on the Armmite L4

SPI3 OPEN speed, mode, bits SPI3 READ nbr, array() SPI3 WRITE nbr, data1, data2, data3, … etc or SPI3 WRITE nbr, string$ or SPI3 WRITE nbr, array() SPI3 CLOSE	See Appendix D of the Micromite User Manual. **Not available on the STM32L432KC**
SYNCH [#]n	Synchronizes an open file on the flash memory. Any pending writes are written out to the flash memory. Can be used anytime a file is open to ensure no data is lost in the event of a power failure.
TEXT x, y, string$ [,justification] [, font] [, scale] [, c] [, bc]	As described in the Micromite User Manual the alignment of the text in the TEXT command can be specified by using one or two characters in a string expression for the third parameter of the command. In the Armmite L4 you can also specify a third character to indicate the rotation of the text. This character can be one of: N for normal orientation V for vertical text with each character under the previous running from top to bottom. I the text will be inverted (ie, upside down) U the text will be rotated counter clockwise by 90º D the text will be rotated clockwise by 90º Positioning is relative to the top left corner of the character when viewed normally so inverted 100,100 will have the top left pixel of the first character at 100,100 and the text will then be above y=101 and to the left of x=101. Similarly “R” in the alignment string is viewed from the perspective of the character in whatever orientation it is in (not the screen).
TIME$	Sets the time in the RTC which is used for all subsequent TIME$ function calls
TRIANGLE X1, Y1, X2, Y2, X3, Y3 [, C [, FILL]]	Draws a triangle on the LCD display panel with the corners at X1, Y1 and X2, Y2 and X3, Y3. 'C' is the colour of the triangle and defaults to the current foreground colour. 'FILL' is the fill colour and defaults to no fill (it can also be set to -1 for no fill). All parameters can now be expressed as arrays and the software will plot the number of boxes as determined by the dimensions of the smallest array. x1, y1, x2, y2, x3,and y3 must all be arrays or all be single variables /constants otherwise an error will be generated c and fill can be either arrays or single variables/constants.
WS2812 pinno, colours%()	This command outputs the required signals needed to drive WS2812 LED drivers on the pin specified. The colours%() array should be sized to have exactly the same number of

	elements as the number of LEDs to be driven. Each element in the array should contain the colour in the normal RGB888 format (0 - &HFFFF) e.g. *dim b%(6)=(rgb(red), rgb(green), rgb(blue), rgb(Yellow), rgb(cyan), rgb(magenta), rgb(white))* *setpin 1,dout* *ws2812 1,b%()* will output the specified colours to an array of 7 WS2812 LEDs

Functions (Armmite L4 additions and differences)

ATAN2(y, x)	Returns atan2 of angles x and y expressed in radians. The atan2 function calculates one unique arc tangent value from two variables y and x, where the signs of both arguments are used to determine the quadrant of the result
BIN2STR$(type, value [,BIG])	This function returns a string containing the binary representation of 'value'. Valid values of "type" are: INT64: returns an 8 byte string containing a signed 64-bit integer UINT64: returns an 8 byte string containing a unsigned 64-bit integer INT32: returns a 4 byte string containing a signed 32-bit integer UINT32: returns a 4 byte string containing an unsigned 32-bit integer INT16: returns a 2 byte string containing a signed 16-bit integer UINT16: returns a 2 byte string containing an unsigned 16-bit integer INT8: returns a 1 byte string containing a signed 8-bit integer UINT8: returns a 1 byte string containing an unsigned 8-bit integer SINGLE: returns a 4 byte string containing a single precision floating point number DOUBLE: returns an 8 byte string containing a double precision floating point number By default the string contains the number in little-endian format. i.e. the least significant byte is the first one in the string. Setting the third parameter to 'BIG' will return the string in big-endian format i.e. the most significant byte is the first one in the string In the case of the integer conversions, an error will be generated if the 'value' cannot fit into the 'type' e.g. an attempt to store the value 400 in a INT8 This function makes it easy to prepare data for efficient binary file I/O or for preparing numbers for output to sensors and saving to flash memory. See also the function STR2BIN
DATETIME$(n)	Returns the date and time corresponding to the epoch number n (number of seconds that have elapsed since midnight GMT on January 1, 1970). The format of the returned string is "dd-mm-yyyy hh:mm:ss". Use the text NOW to get the current datetime string, i.e. ? DATETIME$(NOW)
DAY$(NOW) DAY$(anydate$)	Returns the current day of the week as a string e.g."Monday" Returns the day of the week as a string e.g."Monday" for

	anydate$ which must be specified as “dd-mm-yyyy” or “dd-mm-yy”. The second form assumes 2000 + yy. You can also use ‘/’ as the separator.
DATE$	Reads the RTC and returns the date as a string in the form “dd-mm-yyyy”
DIFF(adcpin1, adcpin2)	Returns the differential voltage between two pins set up as analogue inputs. The voltage will be in the range –VCC to VCC. In the chip pinout listing ADC channels are identified with a channel number e.g. ANALOG_IN_1. The pin pairs must start with an odd number channel and then the next pin must be the even numbered channel one greater. As an example for the 64-pin chip: SETPIN 8,AIN SETPIN 9,AIN PRINT DIFF(8,9) Both voltages on the pins must be in the absolute range 0 to VCC as with any normal analogue input
DIR$(fspec) or DIR$( )	Will search the flash memory chip 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. The function will return the first entry found. To retrieve subsequent entries use the function with no arguments. ie, DIR$( ). The return of an empty string indicates that there are no more entries to retrieve. This example will print all the “.bas” files on the flash chip: f$ = DIR$("*.bas") DO WHILE f$ <> "" PRINT f$ f$ = DIR$() LOOP
DISTANCE(trigger, echo) or DISTANCE(trig-echo)	Measure the distance to a target using the HC-SR04 ultrasonic distance sensor. Four pin sensors have separate trigger and echo connections. 'trigger' is the I/O pin connected to the "trig" input of the sensor and 'echo' is the pin connected to the "echo" output of the sensor. Three pin sensors have a combined trigger and echo connection and in that case you only need to specify one I/O pin to interface to the sensor. The I/O pins are automatically configured by this function and multiple sensors can be used on different I/O pins. The firmware will automatically compensate for CPU speed but slow speeds may yield inaccurate results so it is recommended to set the CPU speed to 80 before executing the command and the resetting it after execution has completed.

EPOCH(DATETIME$)	Returns the the epoch number (number of seconds that have elapsed since midnight GMT on January 1, 1970) for the supplied DATETIME$ string. The format for DATETIME$ is “dd-mm-yyyy hh:mm:ss”. Use NOW to get the epoch number for the current date and time, i.e. ? EPOCH(NOW)
FIELD$(string$, num, sep$)	Parses string$ and returns field ‘num’ when any of the characters in ‘’sep$’ are used to split the field. e.g. PRINT FIELD$("aaa,bbb,ccc", 2, ",") Will return “bbb”
FILESIZE(fname$)	Returns the size of the file fname$. Note the filesize is correct as of the last time the file was closed or saved or the SYNCH command was executed. Note also, if an existing file is opened for output then FILESIZE will give the size of the previous version of the file until the file is closed or SYNCH is called.
LCOMPARE(array1%(), array2%())	Compare the contents of two long string variables array1%() and array2%(). The returned is an integer and will be -1 if array1%() is less than array2%(). It will be zero if they are equal in length and content and +1 if array1%() is greater than array2%(). The comparison uses the ASCII character set and is case sensitive.
LGETSTR$(array%(), start, length)	Returns part of a long string stored in array%() as a normal MMBasic string. The parameters start and length define the part of the string to be returned.
LINSTR(array%(), search$ [,start])	Returns the position of a search string in a long string. The returned value is an integer and will be zero if the substring cannot be found. array%() is the string to be searched and must be a long string variable. Search$ is the substring to look for and it must be a normal MMBasic string or expression (not a long string). The search is case sensitive. Normally the search will start at the first character in 'str' but the optional third parameter allows the start position of the search to be specified.
LLEN(array%())	Returns the length of a long string stored in array%()
MM.BOOTCOUNT	When flash memory is enabled, the system keeps a record of the number of times the processor has been restarted in a special file on the flash memory device. This count can be read by using the MM.BOOTCOUNT function. The count is zeroed if the flash memory chip is re-“FORMAT”ed
MM.DEVICE$	Returns “Armmite L43x – nn pin”, where nn is the number of pins on the chip.
MM.FREE	When flash memory is enabled, returns the number of free bytes on the device.

MM.INFO$(AUTORUN)	Returns “On” or “Off” depending on the status of OPTION AUTORUN
MM.INFO$(CPUSPEED)	Returns the CPU speed as a string
MM.INFO$(LCDPANEL)	Returns the name of the LCD panel configured or a blank string
MM.INFO$(PIN pinno)	Returns the status of I/O pin “pinno”. Valid returns are: “Invalid”, “Reserved”, “In Use”, and “Unused”
MM.INFO$(FLASH)	Returns the status of the Flash memory file system. Valid returns are: “Disabled”, and “Ready”
MM.INFO$(TOUCH)	Returns the status of the Touch controller. Valid returns are: “Disabled”, “Not calibrated”, and “Ready”
PIN(BAT)	Returns the voltage being supplied on the VBAT pin to the processor.
PIN(TEMP)	Returns the core temperature of the processor in degrees centigrade
PIN(SREF)	Returns the stored value of the internal reference voltage when the chip was calibrated during production and VCC was 3.0 volts. The internal reference voltage is nominally 1.2V
PIN(VDDA)	Returns the value of the external analogue reference voltage (VREF+/VDDA pin) by using the known value of the internal reference. Use: OPTION VCC pin(VDDA) to set analogue measurements to accurately compensate for the actual value of the external reference
SPI2(n)	See Appendix D of the Micromite User Manual. This function is not available if flash memory and/or a SPI display is configured (with or without TOUCH) on the Armmite L4
SPI3(n)	See Appendix D of the Micromite User Manual. Not available on the STM32L432KC

STR2BIN(type, string$ [,BIG])	This function returns a number equal to the binary representation in ‘string$’. Valid values of ‘type’ are: INT64: takes an 8 byte string containing a signed 64-bit integer, returns an MMBasic INTEGER UINT64: takes an 8 byte string containing an unsigned 64-bit integer, returns an MMBasic INTEGER INT32: takes a 4 byte string containing a signed 32-bit integer, returns an MMBasic INTEGER UINT32: takes a 4 byte string containing an unsigned 32-bit integer, returns an MMBasic INTEGER INT16: takes a 2 byte string containing a signed 16-bit integer, returns an MMBasic INTEGER UINT16: takes a 2 byte string containing an unsigned 16-bit integer, returns an MMBasic INTEGER INT8: takes a 1 byte string containing a signed 8-bit integer, returns an MMBasic INTEGER UINT8: takes a 1 byte string containing an unsigned 8-bit integer, returns an MMBasic INTEGER SINGLE: takes an 4 byte string containing single precision floating point number, returns an MMBasic FLOAT DOUBLE: takes an 8 byte string containing single precision floating point number, returns an MMBasic FLOAT By default the string must contain the number in little-endian format. i.e. the least significant byte is the first one in the string. Setting the third parameter to ‘BIG’ will interpret the string in big-endian format i.e. the most significant byte is the first one in the string. This function makes it easy to read data from binary data files or interpret numbers from sensors or efficiently read binary data from flash memory chips. An error will be generated if the string is the incorrect length for the conversion requested See also the function BIN2STR$
TIME$	Reads the RTC and returns the time as a string in the form “hh:mm:ss” or “hh:mm:ss.sss” if OPTION MILLISECONDS ON is set.