PicoMite MMBasic V7.00.00: what has changed since V6.03.01

This document is for anyone running V6.03.01, the last full release (4 August 2026), or V6.03.00 (the few changes V6.03.01 made are listed at the end). It describes what is different in V7.00.00: what is new, what has changed, above all anything that can make an existing program behave differently, and which faults of V6.03.01 are fixed. Each feature is described as it is in V7.00.00. The release notes on GitHub keep the history release by release, for anyone who followed the betas and release candidates.

Version 7 makes MMBasic faster on every board, and on the RP2350 it adds a compiler that runs most of a program as native code while leaving the program exactly as you wrote it: switch it on with OPTION COMPILE ON, type RUN, and whatever the compiler can take runs compiled while the rest runs as before. Programs get more memory, a whole array can be copied in one assignment and a function can return an array, a program can load its own library from files, images can be kept in PSRAM, sprites are drawn over a background on VGA and HDMI, and a BASIC routine can be compiled to a CSUB without writing any C. The USB host software has been replaced, which makes hubs, USB drives and USB serial adapters faster and more reliable; the Bluetooth host builds add Classic keyboards, speakers and headphones, and serial devices; and a HELP file, Print Screen, SAVE PNG, ZMODEM, BBC Micro sound and listings to a printer, serial port or file have arrived.

There are two new builds. The PicoMitePC3 puts the HDMI PicoMite's video and USB, the WebMite's WiFi and the Bluetooth host's keyboards, speakers and serial devices in one image for the Pico Computer 3, and also runs the Pico Computer 2 and the Olimex RP2350pc. The PicoMiteHDMIBTH brings Bluetooth keyboards, mice and speakers to the HDMI PicoMite. The release has 18 builds in all:

RP2040	**RP2350**
PicoMiteRP2040, PicoMiteRP2040USB, PicoMiteRP2040MIN	PicoMiteRP2350, PicoMiteRP2350USB, PicoMiteRP2350VGA, PicoMiteRP2350VGAUSB
PicoMiteRP2040VGA, PicoMiteRP2040VGAUSB	PicoMiteHDMI, PicoMiteHDMIUSB, PicoMiteHDMIWEB, PicoMiteHDMIBTH (new)
WebMiteRP2040	WebMiteRP2350, PicoMiteRP2350BT, PicoMiteRP2350BTH, PicoMitePC3 (new)

MM.VER reads 7.000099. A program that compares MM.VER with an earlier release, such as MM.VER >= 6.0302, is unaffected.

Upgrading from V6.03.01

Every board is reset when it first boots V7.00.00. Back up first. On the first boot the options return to their defaults, saved variables and the three flash slots are erased, program memory is cleared and the A: drive is reformatted. Copy anything on A: to an SD card or your PC, and save your settings while the board still runs V6.03.01:

```
OPTION DISK SAVE "B:/myoptions.txt"
```

Save the file to B:, not A:. After upgrading, put the settings back with OPTION DISK LOAD "B:/myoptions.txt", then check them with OPTION LIST: see the next section.

Options saved by V6.03.01

V6.03.01's OPTION DISK SAVE had faults, which V7.00.00 fixes but cannot undo in a file it has already written. After OPTION DISK LOAD of a file saved by V6.03.01:

- On every build but the PicoMiteRP2040MIN, every positive option held as a decimal number lost its first digit: a mouse sensitivity of 2 comes back as 0 (off) and 1.5 as 0.5. Give OPTION MOUSE SENSITIVITY again. The touch calibration is stored the same way and came back right only when its scales were negative or between 0.0001 and 1, so calibrate the touch screen again if it is off.
- The platform name and the telnet console mode were not saved: set them again if you use them.
- On the PicoMiteHDMIWEB the mouse sensitivity was not saved at all.

A file saved by V7.00.00 has none of these faults.

After the first boot

- A Pico Computer 3 configures itself again at the first boot; its CPU speed and any options you set by hand need giving again. A board that uses a preset can simply be given it again (OPTION RESET name).
- On an RP2350 board with 16 MB of flash (a Pico Computer 3 or a Pimoroni Pico Plus 2, for example) the A: drive now has a fixed place, 12 MB from 4 MB to the end of the flash, the same in every RP2350 build. From now on, flashing a different build, OPTION RESET and the presets keep the A: drive and its files. On these boards the 1 MB MOD buffer is always there, and OPTION MODBUFF is not used.
- From V7.00.00 on, a later firmware of the same build resets nothing unless its release notes say so.
- UPDATE FIRMWARE is in every build, so the USB host builds can reach the bootloader without the BOOTSEL button.

Program size and memory by build

The largest program each build takes, the heap MMBasic has for variables, arrays and strings, and the number of local and global variables:

Build	Largest program	MMBasic heap	Locals / globals
PicoMiteRP2040	120 KB	152 KB	240 / 256
PicoMiteRP2040USB	120 KB	152 KB	240 / 256
PicoMiteRP2040MIN	112 KB	156 KB	240 / 256
PicoMiteRP2040VGA	100 KB	120 KB	240 / 240
PicoMiteRP2040VGAUSB	96 KB	116 KB	240 / 240
WebMiteRP2040	88 KB	108 KB	192 / 240
PicoMiteRP2350	208 KB	364 KB	192 / 576
PicoMiteRP2350USB	208 KB	360 KB	192 / 576
PicoMiteRP2350BT	208 KB	336 KB	192 / 576
PicoMiteRP2350BTH	208 KB	320 KB	192 / 576
WebMiteRP2350	208 KB	312 KB	192 / 576
PicoMiteRP2350VGA	168 KB	228 KB	192 / 544
PicoMiteRP2350VGAUSB	160 KB	220 KB	192 / 544
PicoMiteHDMI	160 KB	220 KB	192 / 544
PicoMiteHDMIUSB	152 KB	212 KB	192 / 544
PicoMiteHDMIWEB	156 KB	208 KB	192 / 544
PicoMiteHDMIBTH	180 KB	220 KB	192 / 544

The PicoMitePC3 takes a 256 KB program. On most builds the A: drive is smaller than it was; on an RP2350 board with 4 MB of flash it keeps between 1.1 and 1.8 MB, and on a board with 16 MB it is 12 MB.

Changes that can affect existing programs

Each change below can make a program that ran on V6.03.01 stop with an error, give a different result or no longer fit in memory; where there is something to change, the item says what. Several of them are MMBasic now enforcing a rule the manual has always given.

Programs that now stop with an error

- /* and */ are checked when a program is loaded. As the manual has always said, they must be the first thing on their line (after a line number, if it has one) and be followed by a space or the end of the line, and a /* cannot be used inside a SUB or FUNCTION. A line that breaks one of these rules is refused when the program is loaded or the line is typed ("/ and / must be followed by a space or end of line", "/ and / must be at the start of a line" or "/* cannot be used inside a SUB or FUNCTION"), and the program is not loaded at all. Write a banner as /* ***** rather than /*****, and use ' comments inside a routine; a block between routines is fine.
- Text after a SUB's bracketed arguments. A SUB called with its arguments in brackets ignored anything after the

closing bracket: MySub (a - b) \ 8 - 1 passed a - b, and MySub (a), b dropped b. That text is now "Unexpected text". Put the whole argument in the brackets: MySub((a - b) \ 8 - 1). A FUNCTION call is not affected.
- Duplicate labels. The same label twice in a program, or in both the program and the library, is now the error Duplicate label when the program is run; the second used to be silently ignored. Rename one of them.
- A variable with the name of a SUB or FUNCTION now stops a program on the RP2040 with A sub/fun has the same name, as it always has on the RP2350. Rename one of the two. On the RP2350 a FUNCTION called as a command, or a SUB used in an expression, is now reported at the call, as on the RP2040.
- Names of at most 31 characters. Variable, label, SUB and FUNCTION names are limited to 31 characters, as the manual has always said; a 32-character name was accepted and is now refused. Shorten it.
- Lines too long once tokenised. A line that tokenises to more than 255 bytes (a long line of short assignments, say) is refused with Line is too long, instead of overwriting memory, which could restart or hang an RP2040. Split the line.
- Commands that go on using a variable after their statement has finished now refuse a LOCAL with Must be a global variable, because a local is freed when its SUB returns while the command goes on using its memory: IR, KEYPAD, PLAY STREAM, PLAY ARRAY, PLAY SAMPLE, PLAY LOAD SOUND, WEB TCP CLIENT STREAM and its request string, PIO DMA RX and TX, ADC RUN, ADC START with an interrupt, MATH PID INIT and GUI LISTBOX. Use a global (it can be passed down to the SUB) or a STATIC.
- Text that used to be ignored. MM.INFO(FONT), MM.INFO(FLASH), MM.INFO(ADC) and MM.INFO(BOOT) stop with "Unexpected text" if anything follows the keyword (a misspelt MM.INFO(FONT ADRESS 1) used to give the current font's number), and so do MM.INFO(FONT ADDRESS n) and MM.INFO(FONT POINTER n) if anything follows n. BLUETOOTH STATUS and STEPPER STATUS stop with an error if anything but #n follows them; during a stepper job that error does the safe stop, as any error does. LOAD's suffix letters are matched in full, so LOAD "p.bas", RUN is a syntax error rather than ,R: write LOAD "p.bas", R.
- BOUND() raises an error for a dimension the array does not have, or on a variable that is not an array, where it used to return 0. A program that tests an array's rank with If Bound(a(), 2) = 0 needs changing.
- Float overflow, infinity and NaN. A float subtraction that overflows (1e308 - (-1e308)) stops with the same error as addition and multiplication instead of giving INF. Converting NaN, infinity or anything beyond +-2^63 to an integer raises Number too large, including through INT and FIX, which used to return +-9223372036854775807. Minus infinity now prints as -INF and NaN as NAN.
- The USB drive C: with no drive plugged in. OPEN, EXISTS FILE, EXISTS DIR and every other use of C: stop with "USB drive not found", where they stopped with a bare "Error" and an empty MM.ERRMSG$, and so does MM.INFO(EXISTS DIR "C:/"), which used to give 1. Test MM.INFO$(USBDRIVE) first to avoid the error.
- WebMite: TLS certificates while a TLS connection is open. WEB TLS CA and WEB TLS NOVERIFY freed the settings an open WEB OPEN TLS CLIENT or STREAM, or a WEB MQTT CONNECT on port 8883, was still using. While such a connection is open they now stop with "Close the TLS client connection first" or "Close the MQTT connection first". Load the certificates before opening the connection.

Commands and options that have gone

- OPTION TRACECACHE, OPTION CACHE DEBUG and OPTION CACHE SUB have been removed: the statement cache rarely paid for itself and could make programs slower. A program that still uses one of them stops at that line with Invalid Option: delete the line.
- PEEK(VARTBL, n) and POKE VARTBL have gone, because the variables are no longer kept in one table. They stop with an error that points to PEEK(VARHEADER var), which still gives a variable's details. For CSUB writers, call-table entry 0x60 is now g_slotrec, the record of each variable number, in place of the table itself (see PicoCFunctions.h).
- PicoMiteRP2040MIN: no MATH SENSORFUSION. It has been left out to give this build room; every other build keeps it.
- PicoMiteRP2350BTH: I2S and Bluetooth audio only. This build no longer has PWM audio (OPTION AUDIO pin, pin), the SPI DAC (OPTION AUDIO SPI) or the VS1053 (OPTION AUDIO VS1053 and the PLAY commands that need it, such as PLAY MIDI and PLAY STREAM), and the PicoMiteHDMIBTH has none of them either. A PWM or SPI audio option set by an earlier firmware is ignored: set OPTION AUDIO I2S or OPTION AUDIO BLUETOOTH instead.

Different results

- -2^2 is -4. A minus sign in front of a value now applies after the powers that follow it, as in mathematics: -2^2 is -(2^2), -4, and EXP(-x^2/2) is the Gaussian. It used to be (-2)^2, 4. A minus after ^ still belongs to the exponent (2^-2 is 0.25), and ^ still works left to right (2^3^2 is 64, -2^3^2 is -64). Write (-2)^2 for the old result.
- MATH RANDOMIZE and MATH(RAND) seeded their Mersenne Twister with the wrong multiplier (6069 instead of Knuth's 69069). They now give the standard generator's sequences, so a given seed gives other numbers than before.
- MATH(CROSSING ...) returns its index with OPTION BASE applied, so under OPTION BASE 1 it agrees with the array's numbering.
- CHAIN keeps the font and colours on the VGA and HDMI builds, as FLASH CHAIN and RAM CHAIN do; it used to put the default font and colours back. If the font was the last program's own and the next program does not have it, the default font is used. CHAIN at the command prompt starts a program, as RUN does.

Memory and program size

- The symbol tables take memory. V7's speed comes from tables built each time a program is run (see Speed, memory and the compiler): about 20 bytes for each name plus a fixed part, typically 3 to 5 KB. On an RP2350 with PSRAM most of that is put in the PSRAM; on an RP2040, or an RP2350 without PSRAM, it comes out of the program's own memory. Every build has more heap than before, so most programs fit, but one that used nearly all of its memory (a framebuffer, large arrays, a PIO ring buffer, sound buffers) can now stop with Not enough memory. When the tables would have covered the request the message says so, Not enough memory for 1920 bytes: put OPTION SYMBOLS OFF first: then put OPTION SYMBOLS OFF at the top of the program. Try the program without it first.
- Making a variable can fail. Variables now take their memory from the heap, so making one can fail with Not enough memory once the program has used up the heap, just as DIM of an array always could. Room for the first 32 global and 32 local variables is set aside when the program starts, so small programs never meet this.
- Programs of up to 208 KB on five RP2350 builds. The PicoMiteRP2350, PicoMiteRP2350USB, PicoMiteRP2350BT, PicoMiteRP2350BTH and WebMiteRP2350 take programs of up to 208 KB rather than as large as their heap, because the PSRAM the firmware reserves has to hold the five RAM slots and three program sizes of the compiler's stream. Every build's maximum program size is now fixed, so the flash layout and the A: drive no longer move when the heap grows. To fit a large program, LOAD file$, C crunches it as it loads, as AUTOSAVE C and XMODEM C already could.
- OPTION LOCAL VARIABLES applies to the program that sets it. RUN, NEW and LOAD now return to the default split rather than leaving it in force for the next program; CHAIN keeps it.

Speed, memory and the compiler

V7.00.00 runs programs faster on every board and gives them more memory. On the RP2350 it adds a compiler, OPTION COMPILE ON, that runs most of a program as native code while the program itself stays exactly as you wrote it.

New and changed

A faster interpreter on every board

- Names are stored as symbols. When a program is saved, every variable, SUB, FUNCTION and label name becomes a two or three byte symbol, and the interpreter finds what a symbol refers to directly instead of searching for the name each time. LIST and EDIT show every name exactly as you typed it, and a program saved to a file is plain text as before. The tables that link the symbols to what they name take a few KB of memory (see OPTION SYMBOLS OFF below).
- Local variables live on a stack. A SUB's or FUNCTION's parameters and LOCAL variables are made on entry and dropped on exit without searching or tidying a table, and each SUB's parameter list is read once rather than on every

call; on the RP2350 integer and float parameters, a FUNCTION's result and its LOCAL integers and floats are made without searching for their names at all. Loops of calls are 12% to 39% quicker on a PicoComputer 3 and 17% to 55% on an RP2040. Parameter lists also take less memory: a program that calls 40 SUBs of two parameters each needs about 3 KB for them instead of 11 KB. LOCAL strings and arrays are made faster when the heap is full.

- One index for SUBs, FUNCTIONs, CSUBs, labels and line numbers. When a program is made ready to run, MMBasic builds a small index of the SUB, FUNCTION, CSUB and label names in the program and the library, and of the line numbers it jumps to (after GOTO, GOSUB and RESTORE, after THEN, ELSE or GOTO in an IF, and in ON ... GOTO and ON ... GOSUB lists). So a name found by its spelling (a command typed at the prompt, EXECUTE, CALL, ON ... GOTO, RESTORE, a program run with OPTION SYMBOLS OFF) and a jump to a numbered line no longer search the program. A target worked out while the program runs (GOTO 100 + n) is searched for as before. On the RP2350 the index replaces a fixed 18 KB table, and takes a few hundred bytes for most programs.
- Quicker RP2040 PicoMiteVGA, PicoMiteVGAUSB and WebMite. These builds keep much of MMBasic in flash; finding a variable, calling a SUB or FUNCTION and IF now run from RAM, and on every RP2040 build so does double-precision arithmetic.
- Interrupts. With an interrupt armed (a SETTICK, a pin, a COM port...) the interpreter used to check every source after every statement. Each source now signals when it is ready, so an armed but quiet interrupt costs almost nothing: a statement loop with SETTICK armed takes about 30% less time on the PicoComputer 3 and the RP2040 PicoMiteVGA.
- DO loops. The fast path for DO loops is on every build: a DO WHILE i% < 100000 loop on the RP2040 PicoMiteVGA went from 1.50 s to 0.60 s.
- MATH FFT runs about 20% faster and MATH FFT INVERSE about 25%: a 1024-point transform takes 10.4 ms instead of 12.9 ms, and its inverse 12.4 ms instead of 16.7 ms (WebMiteRP2350). Results are unchanged; only a complex multiplication or division of infinite or not-a-number values can now give NaN parts where it gave an infinity.
- How much faster. Against V6.03.01 on an RP2040 PicoMiteVGA at 378 MHz, PETSCII Robots updates a frame in 54.6 ms instead of 64.7 ms (16% less) and Solar eclipse runs in just under 23 s instead of 28 s (about 18% less). These were measured early in V7's development, before several of the changes above.

More memory for programs

- Variables live in the heap. MMBasic used to set aside a fixed block of memory for every variable a program could ever have, whether it used five or five hundred: 27 KB on the RP2040 and up to 49 KB on the RP2350. That block has gone and its RAM is now heap. A variable takes memory from the heap only when the program creates it, in groups of 32 (1.75 KB on the RP2040, 2 KB on the RP2350), so a program with 40 variables uses memory for about 40. MEMORY shows the variables' share as it grows.
- More heap on every build, for example 120 KB on the RP2040 PicoMiteVGA and 364 KB on the RP2350 PicoMite (each build's figure is in the table under Upgrading from V6.03.01). That is room for bigger arrays, more strings, sound buffers and framebuffers.
- Local and global variables. The RP2350 builds have 192 local entries and 544 or 576 globals; the RP2040 builds have 240 locals, except the WebMite, which has 192, and 240 or 256 globals. Real programs use far fewer: the most we have measured is 125. local_peak in the OPTION PROFILING report shows how many a program uses, and OPTION LOCAL VARIABLES moves the split as before.

Programs that need all their memory: OPTION SYMBOLS OFF

- What it is for. OPTION SYMBOLS OFF runs one program without the symbol tables, and gives the program the memory they would have taken (about 20 bytes for each name plus a fixed part, typically 3 to 5 KB). Use it only for a program that runs out of memory. You do not have to guess: when a program runs out of memory for a request no larger than the tables, the error message says so:

```
Error : Not enough memory for 1920 bytes: put OPTION SYMBOLS OFF first
```

- Using it. Put OPTION SYMBOLS OFF on a line of its own at the top of the program. Only blank lines, comment lines and a LIBRARY LOAD may come before it, and a LIBRARY LOAD can equally come after it. Only that program is affected: nothing is saved in the options. The program itself does not change (it is still stored with symbols, and

LIST, EDIT and SAVE behave as before), so the line can be added or removed at any time. At the command prompt it gives Invalid at command prompt, and lower down in a program Must be at the top of the program.

- What it costs. Speed: every name is found as in earlier versions, and a little more slowly because it is stored as a symbol. In the interpreter's 36 micro benchmarks on an RP2040 PicoMiteVGA at 315 MHz, V7 with its tables takes 21% less time per statement than V6, and without them 4% more. The largest differences are in calls to a SUB or FUNCTION with parameters (a SUB with six parameters and four LOCALs takes 89 us with the tables, 138 us without them and 122 us in V6), in GOSUB (4.5 us against 39 us) and in long names. On an RP2350 a program with the line is not compiled.
- Most programs no longer need it, now that every build has more heap. A logic analyser that needed it on an RP2040 PicoMiteVGA now runs with its symbols, and its main loop takes 81 ms instead of 110 ms.

The compiler (RP2350)

On the RP2350 builds MMBasic can compile a program when it is run. To try it, type OPTION COMPILE ON at the command prompt and RUN your program; afterwards PRINT MM.INFO(COMPILE) shows how much of it ran compiled. If the program behaves differently from the interpreter, run it once under OPTION COMPILE SHADOW to find the line.

```
OPTION COMPILE ON        ' compile at the next RUN
OPTION COMPILE SHADOW    ' compile, and check every compiled value against the interpreter
OPTION COMPILE OFF       ' interpret only (the default)
```

- Setting it. OPTION COMPILE is typed at the command prompt and takes effect at the next RUN. It is saved with the other options, so it stays in force after a restart until you change it; OPTION LIST shows it when it is ON or SHADOW, and OPTION DISK SAVE and OPTION DISK LOAD carry it.
- What happens at RUN. The firmware compiles the program, and its library if there is one, into a stream of native operations held apart from the program. It compiles only when the program or library has changed since the last compile, so the first RUN after an edit takes a moment longer (about 0.3 s for a 110 KB program on a PicoComputer 3) and later runs start as quickly as the interpreter. Statements the compiler cannot take stay in the stream as references to your text and run on the interpreter, between the compiled ones, so a program never has to be entirely compilable to benefit. Nothing about the program changes: LIST, EDIT, SAVE, error messages and line numbers, TRACE, ON ERROR, interrupts, CHAIN and libraries all behave as before.
- When a program runs entirely as text. When it starts with OPTION SYMBOLS OFF (MM.INFO(COMPILE) says TEXT: OPTION SYMBOLS OFF), when it was saved without symbols (TEXT: saved without symbols), and when its stream would not fit the space kept for it (TEXT: too big for the slot (n of m bytes)). A stream is typically 1.4 to 3 times the size of the program, and the space kept is three times the maximum program size, so every program that fits in memory should compile.
- Where the stream is kept. With PSRAM, in the part of the PSRAM the firmware already reserves, above the five RAM slots: none of your PSRAM heap or RAM slots is used, and the stream is lost at power-off (the next RUN compiles again). Without PSRAM, in a hidden area of flash straight after the program, three program sizes long, rewritten only when the program has changed since its last compile, so an edit-and-run costs one erase of it, as saving the program already does. This area is why the A: drive is smaller on the RP2350 builds.
- Memory and recursion. The compiler borrows some temporary memory while it compiles and returns it before the program starts; while the program runs it uses none of the heap. Compiled code recurses as deeply as the interpreter or deeper.
- Speed. Times interpreted and compiled on a PicoComputer 3 (RP2350 at 378 MHz):

program	**interpreted**	**compiled**	
landscape (untyped FUNCTION in the inner loop)	78.5 s	21 s	3.7x
torus (nested loops, IF and PIXEL)	40.5 s	7.8 s	5.2x
galaxy (SIN/COS per point), per point	255 us	80 us	3.2x
reference benchmark	15.4 s	7.1 s	2.2x
Elite, per frame	22.6 ms	11.0 ms	2.1x
Prince of Persia port, per frame	9.7 ms	6.7 ms	1.4x
PETSCII Robots, frame update	36.0 ms	23.4 ms	1.5x

Where the gain is smaller, the time goes on work the compiler cannot make faster: the maths library (a SIN or a ^ costs the same compiled or not), the drawing itself, and statements that are not compiled.

What compiles, in outline

Anything not listed here runs on the interpreter exactly as before. A statement that the compiler cannot take in full runs as text as a whole; the statements around it are still compiled.

- Statements. Assignments to a numeric or string variable or a numeric array element. Multi-line IF ... THEN ... ELSE ... ENDIF with any number of ELSEIFs, and a single-line IF whose THEN and ELSE parts are assignments, or with no ELSE and any statement after THEN (but not THEN GOTO or THEN and a line number). FOR ... NEXT with a numeric loop variable; DO ... LOOP with or without WHILE or UNTIL; SELECT CASE with a numeric selector, with values, ranges (a TO b) and IS comparisons; EXIT FOR, EXIT DO and CONTINUE FOR. Calls to a SUB (Foo a, b or Foo(a, b)) with up to 16 parameters, each argument a numeric expression, a variable or array element (passed by reference) or a constant (passed by value); RETURN, END SUB, EXIT SUB, END FUNCTION and EXIT FUNCTION. LOCAL in every form, and STATIC on every call after the first. INC. BOX, LINE, PIXEL, COLOUR, TEXT, CIRCLE, TRIANGLE, ARC and RBOX, and BLIT x1, y1, x2, y2, w, h, BLIT FLASH, BLIT FRAMEBUFFER, BLIT READ, BLIT WRITE, BLIT CLOSE and BLIT MEMORY (SPRITE MEMORY). Comments, empty lines, labels, DATA, ENDIF, END SELECT and /* ... */ blocks cost nothing.
- Expressions. The operators + - * /, \, MOD, ^ (when its left operand is a float), the comparisons, AND OR XOR, NOT, INV, <<, >> and unary - and +, with + and the comparisons on strings. Decimal, hexadecimal, octal and binary numbers, and strings of up to 64 characters with no backslash. Integer, float and string variables, global or local, including parameters, STATICs and constants; numeric array elements of any number of dimensions; dotted names (R.running) in a program that defines no TYPE.
- Built-in functions. SIN COS TAN ATN SQR EXP LOG DEG RAD INT FIX ABS SGN PI RND ASIN ACOS ATAN2 CINT MAX MIN; LEN ASC CHR$ MID$ STR$ SPACE$ LEFT$ RIGHT$ UCASE$ LCASE$ HEX$ OCT$ BIN$, INSTR and TRIM$; RGB, MAP(n), PIXEL(x, y) and KEYDOWN(n); TIMER, INKEY$, DATE$, TIME$ and the other functions with no argument; the integer MM. values (MM.HRES, MM.VRES, MM.ERRNO and the rest); PEEK of a BYTE, WORD, SHORT, INTEGER or FLOAT address and PEEK(VAR name, offset); MATH(RAND); CHOICE(condition, a, b) when a and b have one type; MM.INFO(...) with the keywords most used in loops (FONTWIDTH, FONTHEIGHT, HEAP, HPOS, VPOS, UPTIME, PATH, CPUSPEED and more); and VAL(s$) as the whole right-hand side of an assignment. They follow OPTION ANGLE and give the same errors as the interpreter.
- User FUNCTIONs that return a number (typed, or untyped and taking the type OPTION DEFAULT gives it), any number of them in an assignment, an IF or ELSEIF condition or an INC value, calls inside other calls' arguments included. Arguments are as for a SUB call.
- Not compiled, among others: PRINT, INPUT, GOTO, GOSUB, ON ... GOTO, DIM, CONST, READ, ERASE, CLS, POLYGON, MATH, SPRITE (other than SPRITE MEMORY), PAUSE, CONTINUE DO, file and pin commands; string array elements, structure members and whole-array assignments; an integer ^; a FUNCTION that returns a string or an array; a call to a CSUB, or a SUB call with a string, array or structure argument; a user FUNCTION call in a SUB's argument, a FOR's start, limit or step, a DO or LOOP condition, a LOCAL value or a drawing command's argument; LINE PLOT, LINE GRAPH, LINE AA and the array forms of the drawing commands; BLIT LOAD, MERGE, RESIZE, COMPRESSED and MEMORY332; VAL anywhere else (its answer is an integer or a float depending on the string), CHOICE whose branches have different types, STRING$, PIN, EVAL and MM.INFO with other keywords. The variables these statements create are used by compiled statements like any others.
- New variables. A variable that does not exist yet is made by the interpreter, as always: the first time a statement that makes it runs, that statement runs as text, and it is compiled from then on. A single-line IF is not held back by a variable that only its THEN or ELSE part uses: IF k$ = "q" THEN quit = 1 runs compiled on every pass but the one that first sets quit.

Known differences from the interpreter

- If a FUNCTION ERASEs a variable that the statement calling it goes on to use after the call, the interpreter quietly makes the variable again; a compiled statement stops with A FUNCTION changed a variable of this statement. A FUNCTION that erases or creates other variables is fine.

- If a FUNCTION changes OPTION DEFAULT while another untyped FUNCTION is still to be called in the same statement, the statement stops with OPTION DEFAULT changed by a FUNCTION in this statement.
- Programs run faster, so a program that relies on how long its statements take (a delay loop rather than PAUSE) will run differently.

Checking and measuring a compiled program

- OPTION COMPILE SHADOW runs the program compiled, and wherever the compiler has worked out a value (the right-hand side of an assignment, an IF, ELSEIF, DO or LOOP condition, a BOX, LINE, PIXEL or COLOUR argument) it works it out again with the interpreter and compares the two. If they differ in any bit the program stops at that line:

```
[120] y = a * Sin(t) + b
Error : SHADOW: compiled 3.5, text 3.25
```

It is slower than ON, since every checked value is worked out twice, and switching between ON and SHADOW recompiles at the next RUN. A statement that calls a user FUNCTION or uses RND, TIMER, INKEY$, KEYDOWN(, MATH(RAND) or MM.INFO(is not checked, because the interpreter would make the call again or get another value.

- MM.INFO(COMPILE) says what the last RUN did. Print it at the end of the program, or type PRINT MM.INFO(COMPILE) at the prompt afterwards:

```
COMPILED 3 REUSED 1 STMTS 3227 RAN 714382 MISS 25707 CODE 680945 SIZE 132348 OF 966656
```

field	**meaning**
COMPILED n / REUSED n	how many compiles, and how many runs that found the stream already compiled, since power-on. At the prompt the word is NONE instead of COMPILED.
STMTS n	statements in the stream
RAN n	statements the last run executed through the stream
CODE n	how many of those ran as compiled code
MISS n	internal: lookups of where to go next
SIZE n OF m	bytes of stream, of the space kept for it

CODE divided by RAN is the share of the work done compiled: 680945 of 714382 is 95% here. It can also say OFF, or TEXT: and a reason (above).

- OPTION PROFILING ON shows which statements ran as text. Put it as the program's first line (it must be in the program: RUN clears it), make sure the program finishes with END (the report is printed by END, not when a program runs off its last line or is stopped with Ctrl-C), and run it with OPTION COMPILE ON:

```
[PERF] elapsed=61247333 us  statements=772063  findvar=...  user_subs=46912  local_peak=19/192
[PERF] top commands by dispatch count:
      325725  Static
      323875  Loop
       56082  Blit Memory
        2172  FRAMEBUFFER
[PERF] top SUBs by exclusive (self) time:
       self_us    incl_us     calls   self_us/call  name
       7444555    7444555       719         10354  writeworld_n
```

With the compiler on, statements and the top commands list count only the statements that ran as text, so the list is exactly what is left uncompiled, command by command, busiest first. The SUB table shows where the time goes, so you can tell whether what is left is in a busy SUB or in start-up code. local_peak is the most local variables alive at once, of the number available.

- OPTION PROFILING ON, SAMPLE also samples where the processor is, and END adds the busiest lines: [PCSLINE] 29 4011 is line 29, numbered as the editor and error messages show it (-1 for a line in the library), with 4011 samples. Together with the command list this points at the lines worth changing.

A typical session:

```
> OPTION COMPILE ON
> RUN
... the program, which starts with OPTION PROFILING ON and ends with END ...
> PRINT MM.INFO(COMPILE)
```

A warning when the stack runs low

When a program has taken the stack to within 1 KB of its limit, the next command prompt shows

```
Warning: the stack came within 536 bytes of its limit, at depth 12
```

where the depth is how many SUBs and FUNCTIONs were running. Deep recursion is the usual cause: a simple recursive FUNCTION reaches about 13 levels on an RP2350 and 9 on an RP2040 before Stack overflow. None of the programs we test with comes near it; if one of yours shows the warning, please send it to us.

Fixed

- An array or string that started exactly at the beginning of the heap (or, on the RP2350, of the PSRAM) was not freed by ERASE, CLEAR, REDIM PRESERVE or the return of a SUB or FUNCTION that had it as a LOCAL. It stayed allocated until the next RUN, and programs that use many strings ran much slower after it (3.4 times in a test). It is now freed.
- IF and ELSE followed by more than one space (ELSE ' comment) scanned their block instead of jumping over it: 2,000 skips of a 40-line block took 84 ms instead of 34. They now jump.
- OPTION PROFILING charged time to the wrong SUB after an interrupt SUB or an ON ... GOSUB; interrupt SUBs now appear in the report under their own name.

The language

Arrays can now be copied whole, returned from a FUNCTION and given a dimension of one element; a program can install its own library, in flash or (RP2350 with PSRAM) in RAM; and a variable can have a command's name. A long list of faults that V6.03.01 had in loops, SUBs, error handling, structures, strings and MATH is fixed.

New and changed

Arrays

- Single-element dimensions. A dimension can hold exactly one element: DIM a(0) under OPTION BASE 0 and DIM a(1) under OPTION BASE 1 used to raise Dimensions, which made a row vector, a column vector or a 1x1 result impossible to declare.

```
Option base 0
Dim rowv(2,0), colv(0,2), dot(0,0)
Math M_MULT rowv(), colv(), dot()    ' a dot product
```

- Whole-array assignment. b%() = a%() copies one array into another in a single memory move. The arrays must be of the same type (and the same LENGTH for strings, the same structure type for structures) and have the same total number of elements; the shapes may differ.
- Functions that return arrays. Function f(n%) As Integer(5) (integer and float only) sets its result by assigning to elements of its own name, and is used in a whole-array assignment, m%() = f(3). It cannot appear anywhere else in an expression. Reading an element of the function's name inside the function is a recursive call, so build the result in a LOCAL array if intermediate values must be read back.

Libraries

- LIBRARY LOAD. LIBRARY LOAD file$ [, O] [, RAM] lets a program install its own library. It must be the program's first statement, and it asks before replacing a library that is already there unless O is given. A hash of the source is kept, so a run that finds the same library already in flash touches nothing. Up to eight files can be named and are

joined in order into one library, and nothing is written until every file has been read. Not on the PicoMiteRP2040MIN.

```
LIBRARY LOAD "graphics.bas", "sound.bas", "maths.bas"
```

- A library in RAM (RP2350 with PSRAM). With the RAM keyword the library goes into RAM slot 5 and takes the place of the flash library, which is left exactly as it was, until END or the next RUN. A program can bring a large library of its own without disturbing the routines you keep in flash. Without PSRAM the keyword is ignored.
- READ reads the library's DATA. When the program's DATA runs out, or it has none, READ falls through to the DATA in the library, and a library routine can read its own DATA.

Variables and SUBs

- A variable can have a command's name. At the start of a statement, a command's name followed by = is now an assignment to a variable of that name, so a program that used the name before the command came still runs: fm% = 0 in a program written before the file manager's FM command, or out% = 0 before the PIO instruction OUT, used to stop with an error. $, % and ! after the name count. Without the = the name is still the command, and an array is not covered: box(2) = 9 is still the BOX command. The exceptions are CASE, FLAGS, REM, RUN and DATA, and the names that are functions or keywords as well (DATE$, TIME$, TIMER, FOR, ELSE ...); labels still cannot be command names.
- Labels no longer count towards the limit on SUBs and FUNCTIONs, and are limited only by memory. On the RP2350 the limit is 512 SUBs, FUNCTIONs and CSUBs (it was 511 SUBs, FUNCTIONs and labels together); the RP2040 keeps 256.

Functions and commands

- MATH CLAMP in(), lo, hi, out() limits every element to a range, integer or float, strided; 16 elements take 14 us.
- BASE$(base, number [, chars]) gives a number in any base from 2 to 36.
- OPTION PICOMITE does nothing on the PicoMite and gives no error, in a program or at the command prompt: OPTION PICOMITE HDMI, OPTION PICOMITE VGA, OPTION PICOMITE OFF, anything else after it, or nothing, is accepted and ignored. A program can include it for the Colour Maximite 2, which has to switch to its PicoMite mode to run a PicoMite program.
- SHIFT with the left and up arrows has key codes of its own, LEFTSEL (&HA2) and UPSEL (&HA4), beside the existing down and right codes.
- MM.INFO$(BOOT) reports a brown-out reset distinctly (RP2350).

Errors

- MM.ERRLINE is the line of the last error, beside MM.ERRNO and MM.ERRMSG$.
- Out of memory names the size, as in Not enough memory for 16000008 bytes. This one message replaces Not enough PSRAM memory, Not enough Heap memory and Not enough System Heap memory, so a program that tests MM.ERRMSG$ for one of those sees different text.

Fixed

Loops, SUBs and program flow

- In a recursive SUB or FUNCTION with a FOR loop around the recursive call, the inner call's NEXT closed the caller's loop too, and the program usually stopped with "Cannot find a matching FOR". With a DO loop around the call it failed with "LOOP without a matching DO". Each call now keeps to its own loops.
- A FOR right after the THEN or ELSE of a single-line IF, inside another loop, was not counted when the outer FOR looked for its NEXT, so the outer loop took the inner NEXT as its own and stopped with "Cannot find a matching FOR"; a DO in a single-line IF inside a DO loop had the same fault. Both are counted now. A NEXT or LOOP after THEN (IF c THEN NEXT i) is still not counted: the loop's own NEXT follows as a statement.
- A FOR, DO or GOSUB whose line ends in : ' comment left the loop after one pass, or returned past the next line.
- On the builds with the trace cache, the DO loop fast path truncated a fractional limit: LOOP WHILE i% < 2.5 stopped

at 2.
- Calling a one-letter SUB alone on a line skipped the next line, and a program containing such a call (a, for SUB a) could fail to save, leaving the board not responding.
- A saved program whose last line is a single-line IF with no ELSE could stop with a fault when the condition was false, for one program length in 256 (in a library or a RAM slot too).
- An error in an ELSEIF condition now names the ELSEIF's line.
- A statement made of two adjacent operator tokens, such as ^^ or IF 1 THEN ^^, was run as a command from past the end of the command table, calling whatever was there. Only a token that names a command is run as one now.
- SUB MM.END did not run when a program ran off its last line, only at an END command, though the manual says it runs at an actual or implied END.
- A SUB with an array parameter that the call gives no argument for, as SUB MM.END(a()) always is, stopped with "Incompatible type:" followed by stray text, often an earlier call's argument; the message is now "Incompatible type: " alone.
- CONTINUE after a break inside a SUB or FUNCTION went on at the program's top level: the SUB's local variables were out of sight (a LOCAL became a new global) and its END SUB then cleared every variable, the globals too.
- After an error inside a SUB or FUNCTION, calling a SUB or FUNCTION at the command prompt failed with "Local variable already declared" until the program was run again.
- A program that stopped with an out-of-memory error could leave the board printing the same error at the prompt for ever, until a hardware reset. It now stops once and gives the > prompt.
- CHAIN could leave the chained program's TYPE definitions and its multi-line IF blocks pointing into the old program's variables; both are now set up again after the kept variables are restored. A chained program that uses structures must define the same TYPEs in the same order as the program that chained to it, as a kept variable of a structure type refers to its type by number.
- A program in a RAM slot could CHAIN once; the next RAM RUN gave "Could not find the file".
- EXECUTE "runs = 5" was run as RUN, and EXECUTE "RUN", or EXECUTE "RUN " + CHR$(34) + "prog" with no closing quote, could crash.

Variables and STATIC

- A STATIC could attach its variable to the wrong SUB: the one called most recently rather than the one running.
- OPTION LOCAL VARIABLES above the default corrupted memory, and it stayed in force after the program that set it had ended, so the next program could run out of global variables. RUN, NEW and LOAD now put it back to the default; CHAIN keeps it for the chained program. A deep recursion could also leave the local variable table unusable ("Too many local variables" with most of it free).
- An expression passed to a BYVAL parameter (Inner 5, x()) could be refused with "Array as BYVAL not allowed" when an unrelated local array happened to occupy the first slot of the variable table. Passing a whole array BYVAL is still refused, and an array element is still accepted.
- After an ERASE had left every global slot used or blocked, looking for a name that was not there could hang at the top level, and inside a SUB gave "Too many global variables" although a slot was free.
- MM.INFO(VARCNT) did not fall after ERASE or when a SUB's locals were freed, and could stay one too high after an error while making a variable, so a later DIM was refused with room to spare.

Arrays

- REDIM recreated arrays as the default type, losing INTEGER and STRING data.
- REDIM overran its buffer when a name and its dimensions reached 32 characters, as REDIM PRESERVE sensor_readings(reading_count% + 1) does.
- DIM s$ LENGTH CHOICE(w = 1, 80, 40) or LENGTH LEN("it's") gave "Expected closing bracket", a bracket in a string in a string array's bounds gave an error or lost the LENGTH, and REDIM PRESERVE buf(BOUND(buf()) * 2), MAX(4, 8) or (n) in REDIM's bounds gave an error. All work now.
- A DIM initial string longer than the variable's LENGTH (scalar, array or structure member) was written in full over what followed; it is now "String too long". A scalar string passed where a string array is required (ARRAY SET "", s$) was taken as an array and overwrote memory; it is refused. DRAW3D CREATE checks that the vertex array is (2,

n).
- An integer array subscript beyond 32 bits is "Index out of bounds"; with DIM a(3), a(4294967297) used to read a(1).
- A call to a FUNCTION with an array argument, used as a graphics coordinate or as I2C, SPI or ONEWIRE WRITE data, gave "Dimensions", and ONEWIRE WRITE pin, 0, 2, a$ + b$ sent a$ alone.
- ARRAY INSERT and ARRAY SLICE named the argument before the array in their errors about it.

Structures

- A structure member used as an array index, as in a(p.x), made the element be read and assigned as if it had the member's type: a float array element read back as the bits of its value, a string element as a number, and an assignment was stored in the wrong form.
- A nested member assigned whole, s.topLeft = r.topLeft, copied the outer type's size over everything after it; a nested member passed to a SUB was checked against the base variable's type, and an integer member was accepted as a structure variable; a path such as a.b(c.d(1).n).e lost its own continuation; and a TYPE typed twice at the prompt was added twice.
- READ pts().x wrote over memory; it is refused with "Struct member arrays not supported for this command". STRUCT LOAD from a file shorter than the structure loaded part of it silently; it is now "Not enough data in the file".
- bag.v(LEN(a$)), a member array indexed with a function, gave "Invalid syntax", and STRUCT SAVE #MAX(1, 0), r gave "Variable name".
- The MATH commands took pts(2).x and one.x as member arrays, so MATH SCALE pts(2).x, ... started at element 2 and ran past the end of the array. Only pts().x is one now ("Argument n must be a member of every element: a().x"), and it gets the same shape check as any other array; MATH(CRC16 ...), AES128 and BASE64 answer "Struct member arrays not supported for this command" rather than "Invalid syntax".
- DIM of a user-defined TYPE could fail with an out-of-memory error reported against a line that was perfectly correct, and adding an unrelated statement elsewhere in the program made it come and go.

Parsing and the tokeniser

- An assignment took the first = in the statement, so a comparison in its brackets (a(CHOICE(k = 1, 1, 2)) = 5) gave "Invalid syntax", as did MID$() and LMID(), and BYTE(), BIT(), FLAG(), PIN(), PORT(), MAP() and CTRLVAL() stored a wrong value. MAP(n) on VGA and SPI LCDs checks n, and SPI LCD MAP RESET no longer prints "Here".
- The rewrites the tokeniser applies to built-in names (BIN$(, MIN(, LCASE$(, LEFT$(, MAX(, HEX$(and others) failed in several ways. A space before the bracket (LEFT$ (a$, 2), MAX (1, 2), HEX$ (255)) gave "not declared"; a line starting with a name such as remaining or remote$ was taken for a REM and got no rewrites, so remaining = Max(3, 7) failed with "MAX is not declared"; and a name that was not rewritten came back in capitals, so a user function MyHex$(was listed as MyHEX$(. A line that the rewrites lengthen past 255 characters overran the line buffer; it is now refused with "Line is too long", whether typed or loaded.
- #DEFINE: one match at the very start of a line made every later match on it ignore the character in front, so #DEFINE "COUNT", "TOTAL" turned account into acTOTAL; #DEFINE "", "X" looped until it ran off the buffer (an empty name now matches nothing); and a #DEFINE string longer than 31 characters is now refused.
- Inside a #COMMENT START ... #COMMENT END block, #DEFINE and #INCLUDE lines still took effect, and a line beginning with a word that starts MMDEBUG, such as a variable MMDEBUGGER, was dropped.
- GUI CLICK PIN pin, INV gave "Invalid syntax"; PLAY SOUND and FRAMEBUFFER WRITE took s$ or l$ as the letter S or L; with OPTION LEGACY ON, CIRCLE took a colour fcol as the fill flag; and GUI FORMATBOX now takes its format in a variable.
- A ' comment after WEB SCAN, MANDELBROT, BLUETOOTH SCAN and a few others was read as an argument.
- A trailing comma in an argument list, such as MATH(CRC8 a(),), could make a command read a stale argument.
- RESTORE takes an expression (CHOICE(), MID$(), a FUNCTION, an array element), and a name that is a label and also an array or variable is taken as the label.

Errors and interrupts

- An ON ERROR IGNORE or ON ERROR SKIP set inside an interrupt routine stayed in force after IRETURN (or the end of an interrupt SUB) when the interrupted code had no ON ERROR of its own, so the main program's errors

were then ignored.
- ON ERROR SKIP was used up by the statements of a FUNCTION that the protected statement called, so an error the statement itself then raised stopped the program. The count is now used up only by the statements at the level where it was given; a SUB's statements still count, as before.
- The ON ERROR setting a program ended with stayed in force at the prompt: after ON ERROR IGNORE every typing error was silent, and after ON ERROR RESTART one reset the board. The prompt now starts without it.
- Under ON ERROR SKIP or ON ERROR IGNORE, an error in a SUB or FUNCTION's arguments leaked memory, about 2.8 KB per error on the RP2040, so a program could run out after a few dozen.
- ERROR "50% off" showed "50536955912 off": %, @, $ and | in the text were taken as format codes.
- The error messages of FORMAT$, STRUCT() and RAY MAP showed stray numbers, and PIO CONFIGURE's showed the wrong base.
- OPTION LOGGING wrote 64 bytes to log.txt for every error: the message, then whatever an earlier, longer message had left behind, and a zero byte after each CR-LF. Each entry is now the message and a CR-LF.
- If an interrupt that ran during a PAUSE ended in an error (or Ctrl-C), the first PAUSE of the next run returned at once, and an interrupt due during it was not called. Each run now starts afresh.
- ON KEY k, sub, MOUSE INTERRUPT ENABLE, GAMEPAD INTERRUPT ENABLE and the Wii Nunchuk and Classic interrupts could fire at once when armed, for a key press or button change from before (a mouse click made before MOUSE INTERRUPT ENABLE, or a key pressed in an earlier program). They now fire only for what happens after they are armed.
- ON KEY k, sub stayed armed after the program ended (END, an error or Ctrl-C), so key k could not be typed at the command prompt until the next RUN or NEW: after a program that used ON KEY 32, the prompt ignored the space bar. It now ends with the program, as SETTICK does.
- A CSUB's interrupt vectors and the SUB set by INTERRUPT survived NEW and RUN, so the next program's tick or pin edge called into the old code. They are now cleared with the rest of the program's state.

Strings and PRINT

- PRINT's comma writes a tab character, but the column PRINT kept for TAB() and POS moved to the next 14-column zone, so after a comma both were wrong: in PRINT 1, 2; TAB(30); "e" the e landed in column 24 on a terminal. The column now moves to the next 8-column stop, where a terminal or a file viewer puts a tab. The screen pads a tab to OPTION TAB (default 2), so there it still differs.
- PRINT #n to a file or a COM port used the console's column: TAB() in a file line counted from wherever the console's cursor was and could break the line with a CR LF, and afterwards TAB() and POS on the console were wrong. Each file and COM port now keeps its own column from the moment it is opened, and the console's is untouched.
- TRIM$ refused where$ given as a lower-case string, such as "r".
- RIGHT$(s$) with one argument read an unset argument instead of saying "Argument count".
- STR2BIN with a string longer than 8 bytes overwrote the stack, which could restart the board, before reporting "String length".
- FORMAT$ with long literal text around a wide field wrote past its 256-byte result before reporting "String too long".
- LONGSTRING LOAD copied the full count even when the string was shorter, past the end of a small array; LONGSTRING LOAD, LEFT, RIGHT and MID ran away with a negative count; and LONGSTRING COPY accepted a source 8 bytes longer than the destination could hold and wrote past the end of the array (it is now "Destination array too small").
- LMID moved the whole array when the replacement changed the length, writing past its end, and allowed a section one byte beyond the string.
- LINSTR's regular-expression form with a start below 1 read before the array and could return a negative position. A start outside the string returns 0, as the plain form does.
- Regular expressions: {n} matched n or more times.
- JSON$ could read 8 bytes past its source array under OPTION BASE 1.
- MM.VER read past the end of the version string on a release build.

Numbers and MATH

- CHOICE took a condition between -1 and 1, such as 0.5, as false.
- MAX and MIN clamped at +-3.4e38, and ABS and SGN did not accept floats beyond 9.2e18.
- An integer ^ takes at most 64 steps: 1^4000000000 used to lock the board for minutes, and Ctrl-C could not stop it.
- RP2040: SIN, COS and TAN in degrees read the wrong argument when it contained a function: SIN(SQR(2.5)) gave sin(2.5 degrees).
- RP2040: a bare RANDOMIZE reported "Number out of bounds" whenever the board had been up for between 36 and 72 minutes (and in alternate spells after that).
- MATH RANDOMIZE 0 made MATH(RAND) return 0 for ever, a seed of 2^31 or more was refused, MATH RANDOMIZE with no argument was refused after 35.8 minutes of uptime, and MATH(RAND) could return exactly 1. A seed of 0 now acts as the default seed, any seed from 0 to 4294967295 is accepted, and MATH(RAND) stays below 1.
- MATH(C_ARG z%), as the manual spells it, was a syntax error; only C_CARG was accepted. Both are now. MATH(C_ADD ...), C_MUL, C_SUB, C_DIV, C_POW, C_CPLX and C_POLAR, and MATH FFT with its MAGNITUDE, PHASE and INVERSE forms, given one argument instead of two used whatever an earlier command had left in its place; they report "Argument count". An integer C_DIV by zero is now trapped ("Divide by zero").
- MATH(CRC8 ...) and MATH(CRC32 ...) with the result reflected applied the end mask before reflecting, unlike CRC12 and CRC16, so an end mask that is not symmetric gave a wrong result. MATH CRC12's masking and the CRC reverseOut argument were wrong too.
- MATH SLICE and MATH INSERT with every index given read before their dimension table and copied from the wrong place; they now stop with "One index must be omitted". They also copied raw bits between float and integer arrays; they convert now.
- MATH ADD and MATH POWER with an integer output rounded before the arithmetic (0.3 + 0.4 gave 0, 16 ^ 0.5 gave 16), and MATH SCALE of an integer array by an integer went through floating point.
- MATH INTERPOLATE refused valid arrays with "Arrays must be different" whenever both inputs were the other type from the output, and MATH SINC with one input point did nothing without a message; it reports "Interpolation needs at least 2 input points".
- MATH WINDOW gave NaN for a constant input (it now gives the output minimum), truncated an integer output, truncated minin/maxin into integer variables (it rounds now), and accepted constants for the min/max variables.
- MATH(MAX) and MATH(MIN) started from +-3e38 instead of element 0.
- MATH V_MULT named argument 1 in errors about its other arrays, Q_MULT and Q_ROTATE named argument 31, and Q_CREATE's size error named argument 4 for argument 5; MATH SHIFT silently ignored a fourth argument other than U (it is now refused); MATH FFT accepted a single element and did not check that the complex array is 2 x n; and MATH PID INIT kept the parameter pointer of a refused call.
- MATH M_INVERSE failed on a 1x1 matrix and leaked a row per call, and determinants leaked heap. MATH CHI indexed past its table above about 8x8 and had wrong degrees of freedom under OPTION BASE 1, and Q_CREATE's angle units were wrong.
- MATH PID START called back for the first time 1000 times too early; it now waits a full sample time.
- MATH AES128 and MATH(BASE64) into a string declared with a short LENGTH wrote past it, corrupting the variable. They report "Too many elements for string output".

Other commands

- PAUSE stopped with "Number too large" for a delay over 2147483 ms (about 36 minutes), although the manual allows 2147483647 ms (about 24 days): PAUSE 3600000, one hour, failed. It now pauses for up to 2147483647 ms; a longer delay is "Number out of bounds", as a negative one is.
- STAR and ASTRO gave "Variable name" when the body was a string function, and ignored the proper motion with RA/Dec out.
- After the reset button or a brown-out, MM.INFO(BOOT) said Restart if the board had last been restarted by CPU RESTART or an OPTION command, and the sign-on banner did not appear. It now gives the real cause, such as Reset Switch.

The editor, file manager and console

Print Screen saves the display from the command prompt, the editor and the file manager; listings can go to a printer, a serial port or a file; HELP has files to read; and the file manager plays a track list. The editor no longer loses pastes, quickly typed keys or the edits when a save fails.

New and changed

- Print Screen. The Print Screen key, or Ctrl-\ on any keyboard or terminal, saves the display at the command prompt, in the editor and in the file manager, as on the Colour Maximite 2, to SCREEN_yyyymmdd_hhmmss.BMP (for example SCREEN_20261025_102910.BMP) in the root of the SD card when one is configured, otherwise of A:. A name that is already used, by two saves in the same second or after the clock has gone back (without an RTC or network time it starts again at every power-up), gets _1, _2 and so on rather than being overwritten. The file is written by SAVE IMAGE, so it works on any display SAVE IMAGE can read; nothing is printed and a failure is ignored, so the editor and the file manager are not disturbed. Ctrl-\ is the byte &H1C, which every keyboard now gives (USB, Bluetooth, PicoCalc and I2C keyboards included, whichever key types \ in its layout) and a serial or telnet terminal with no Print Screen key can send; the Print Screen key's own code is &H9D. On the RP2350 OPTION PRINTSCREEN FORMAT PNG saves SCREEN_yyyymmdd_hhmmss.PNG with SAVE PNG instead, and OPTION PRINTSCREEN FORMAT BMP, the default, goes back to SAVE IMAGE; it is a permanent option, shown by OPTION LIST when it is PNG and kept by OPTION DISK SAVE. A running program receives the two keys as ordinary keys unless it sets OPTION PRINTSCREEN ON: then they save the display after the statement that is running when the key arrives, or at once during PAUSE and INPUT, the program never sees them (INKEY$ and INPUT included), and MM.ERRNO and MM.ERRMSG$ are kept. That option is temporary, off at every RUN and when the program ends, and OPTION PRINTSCREEN OFF turns it off. Not on the PicoMiteRP2040MIN.
- Listings to a printer, serial port or file. LIST ALL, LIST PINS, FILES, OPTION LIST, LIBRARY LIST ALL and FLASH LIST take a file number as their first argument, as PRINT #n does, and send their output to the serial port or file opened as #n instead of the console, with nothing on the screen: no clearing, no font change and no page pauses. LIST ALL #n, file$ sends the file exactly as it is (every line, blank lines and tabs included, unwrapped, each line ending with CR LF), and LIST ALL #n sends the program in memory without wrapping its long lines. FILES #n lists the entries in the order they are stored in the directory, so it cannot sort them ("No sort order with #n"), and unlike FILES on its own at the command prompt it clears neither the variables nor the open files. A program in a flash slot is listed with ALL: FLASH LIST #n, 2, ALL. Not on the PicoMiteRP2040MIN. LIST PINS is now part of the firmware rather than a small BASIC program run behind the scenes; on the console it prints exactly as before. To print a program on a serial printer, for example:

```
SETPIN GP1, GP0, COM1
OPEN "COM1:9600" AS #1
LIST ALL #1, "test.bas"
CLOSE #1
```

- HELP files. The HELP command reads a file called help.txt on the A: drive. Three come with the release, generated from the user manual and the supplementary SPRITE, TILEMAP, RAY, FRAME, STEPPER, STRUCT, DRAW3D, BLIT and GUI manuals, and regenerated for every release so the entry for a command matches the manual beside them. They cover the same topics: help.txt has the full entry, helpmin.txt the syntax and a one-line summary, and helptiny.txt the syntax lines alone. Copy whichever suits the space on your board to A: as help.txt, then type, for example, HELP LIBRARY LOAD. Not on the PicoMiteRP2040MIN, which has no HELP command.
- The file manager plays a track list. Mark audio files in FM with the space bar and press ENTER: when every marked item is an audio file, they play in panel order as a track list, each track starting when the one before it ends. The status line shows Playing 2/5 name and follows the track that is playing; . or > skips to the next track and , or < returns to the previous one, as PLAY NEXT and PLAY PREVIOUS do, and F6 or T stops the list. A list may mix WAV, FLAC, MP3 and MID files and holds at most 20 tracks; MOD files cannot be in one. With any other marks, ENTER acts on the selected item as before. The file manager also shows a cursor at its text prompts (rename, new file, new directory, filter, type-select).
- FM clears memory when it starts. As EDIT does, FM clears the variables of the program that ran last, so the file manager always has the memory it needs. EDIT FILE and the editor in FM switch off IR and KEYPAD if a program left them running, because the editor uses the memory they write to; the program's variables are kept, as before.

Fixed

The editor

- EDIT of a file that ends with a line ending, as most do, showed an empty line after the last one: the cursor could go there, and text typed on it was saved with no line ending. The last line ending is now taken off when the file is read and written back when it is saved, so an unchanged file is saved byte for byte, and a last line with no line ending is given one. An existing empty file opened with a stray character in it, and a second EDIT of a file counted the previous file's lines as well, so Page Down went past the end.
- EDIT of a file that does not exist wrote a 0 at address 0.
- The editor's PgUp, with the top line one line short of a full screen down (a file one line short of a whole number of screens, paged to the end and back), left a blank top row with the text a row lower, and the next PgUp moved the cursor down a line instead of up.
- The editor opened at an error's line (EDIT after an error, or the file manager's E or F4 after a program run from it stopped) showed blank rows below the end of the file when the error was in the last half screen, and when the file had been cut since the error the file manager's E or F4 opened it on a line past the end, where PgDn did nothing and typed text was lost. The editor now opens with the last line no higher than the bottom row, and at the last line when the error's line is no longer in the file.
- EDIT threw away every key that arrived while it redrew the screen, so a paste lost all but the first line, and keys typed quickly after Enter were lost. Now the keys queued during a redraw are dropped only when every one of them is the key that caused it, a held key repeating, so a held arrow still stops when it is let go. A key typed straight after Esc, such as the Y of a quick Esc Y at "Exit and discard all changes", was lost too.
- Esc in EDIT could not discard the changes while a line of a .bas file was over 255 characters: the discard stopped with "LINE n TOO LONG". Only a save checks the length now, and "LINE n TOO LONG" no longer names the wrong line, or line 0, when the long line is the last one .
- After Beautify (F12 or Ctrl-A) EDIT kept the line count from before it, though Beautify adds a blank line after each END SUB and END FUNCTION and can drop blank lines, so the last lines did not scroll into view with Down.
- EDIT with colour coding off (OPTION COLOURCODE OFF, or any file that is not a .bas) turned the display yellow from a /* line to the bottom of the screen.
- EDIT's search (F3, F6, Ctrl-G, Shift-F3) left a match past the right edge off the screen, with the cursor beyond it; the match's line is now scrolled to show it. Esc, F3 or Shift-F3 at one of the editor's prompts after something was typed was added to the text instead of acting on its own, so Esc did not cancel: the search said " NOT FOUND " and an export tried a name with an Esc in it .
- EDIT lost the edits when a save, an F9 import or an F10 / Ctrl-B export failed (a card out, full or write-protected, a directory in the way, an SD card not configured): MMBasic's error ended the editor. The failure is now shown on the message line (" SAVE FAILED: Entry is a dir ", " IMPORT FAILED: SDcard not configured ") and the editor stays open with the text. A save now writes name.$$$ and then renames it, the old file becoming name.bak, so a failure at any point leaves the file or its .bak whole. An import that fills the buffer says how many characters went in, and mark mode's export message is no longer drawn over by the status line.
- The editor misplaced a mouse click on a horizontally scrolled line, and did not refresh its status line after a click from a serial terminal.

The file manager

- With a USB keyboard, holding Up or Down moved the selection one line and stopped. It now keeps scrolling, as with a PS/2 keyboard; PgUp and PgDn still move once per press.
- With the right panel active, its selected row was drawn in the inactive panel's colours after PgUp, PgDn, End or scrolling.
- The file manager froze or faulted copying directories from a USB drive, overflowing its stack, and EDIT from the file manager could then run out of memory.
- The PicoMiteHDMIUSB build crashed launching a program from the file manager.

Listings and the console

- LIST, EDIT and SAVE unfolded the tokeniser's rewrites inside longer names, so a user function TOBASE$(16, x) was listed and saved as TOHEX$(x).
- The console sent the terminal's autowrap escape sequence before every input line; it is now sent once at reset and once when the editor exits.

Graphics and displays

SAVE IMAGE writes much smaller BMP files. On the RP2350 the screen can be saved as a PNG, a PNG of any size loads, image slots can live in PSRAM, and FRAMEBUFFER MERGE puts sprites over a background on VGA and HDMI. A long list of faults in sprites, MODE 1, framebuffers, DRAW3D and the image loaders is mended.

New and changed

- SAVE IMAGE saves smaller files. SAVE IMAGE now saves the smallest exact BMP for the picture, as on the Colour Maximite 2: an area of 16 colours or fewer as a 4-bit BMP and one of up to 256 colours as an 8-bit BMP, each with its palette and RLE compression, and only an area of more colours as 24 bits a pixel, as before. On a PicoComputer 3 a test screen took 5.8 KB in MODE 1 and 20 KB in MODE 3 (154 KB before), and 10 KB in MODE 2, 4 and 5 (38.5 KB and 230 KB), and the saves are quicker; Print Screen's BMP files shrink the same way. LOAD BMP reads all three kinds, as does any image viewer, but a program that decodes the files itself will now meet 4-bit and 8-bit RLE BMPs. The colours saved are the ones shown, after any MAP or MAP SET, and on VGA and HDMI a framebuffer layer is included in every mode, as you see it. SAVE COMPRESSED IMAGE is now the same command, kept for older programs: it works on every display and with any width, where it needed a 16-colour display and an even width.
- SAVE PNG (RP2350). SAVE PNG file$ [, x, y, w, h] saves the screen, or part of it, as a compressed true colour (8-bit RGB) PNG file. It takes the same arguments as SAVE IMAGE, with any width, and adds .PNG when the name has no extension. A 640x480 test screen took 12 KB, against 154 KB for the uncompressed 16-colour BMP. On a 16-colour display the colours saved are the ones shown, after any MAP.
- PNG images of any size, cropped, and in more formats (RP2350). The PNG decoder now works a line at a time, so a PNG needs about 40 KB of memory whatever its size, where it needed room for the whole file and the whole image (2.7 MB for an 800x500 picture), and it is faster: that picture takes 1 second on a 320x240 screen instead of 3. LOAD PNG crops an image larger than the screen, as LOAD BMP and LOAD JPG do, instead of stopping with Image too large, and two more optional parameters give the first column and row of the image to show: LOAD PNG fname$ [, x] [, y] [, transparent] [, alphacut] [, ximage] [, yimage]. SPRITE LOADPNG loads part of an image, as SPRITE LOADBMP does: SPRITE LOADPNG #n, filename$ [, transparent] [, cutoff] [, x_offset] [, y_offset] [, width] [, height]. LOAD PNG, SPRITE LOADPNG and the file manager's viewer also open 24-bit RGB PNGs (RGB888, drawn fully opaque), as SAVE PNG and many PC programs write them, and 8-bit greyscale PNGs with or without alpha, as on the CMM2; before, these reported "Invalid format, must be RGBA8888 or indexed PNG".
- FRAMEBUFFER MERGE on VGA and HDMI (RP2350). FRAMEBUFFER MERGE [colour] [, B] composites framebuffer 2 over the main framebuffer into the display in one pass, without the tearing that drawing into the live layer produces; with B it does so at vertical blank. A full mode 3 merge takes about 11 ms, one frame at 75 Hz. FRAMEBUFFER CREATE 2 shares the video memory the layer would otherwise use, and BLIT FLASH, BLIT FRAMEBUFFER and TILEMAP DRAW accept buffer 2. RP2350 only, as CREATE 2 is.

```
FRAMEBUFFER CREATE          ' the background, F
FRAMEBUFFER CREATE 2        ' the moving things, 2
FRAMEBUFFER MERGE 0, B      ' 2 over F into the display, at vertical blank
```

- Image slots in PSRAM (RP2350). On an RP2350 with PSRAM, image slots 4 to 8 are the RAM slots 1 to 5, and FLASH LOAD IMAGE, BLIT FLASH, TILEMAP and MM.INFO(FLASH ADDRESS) take them by number. A RAM image loads in milliseconds, costs no flash wear and leaves the three flash slots free. RAM LIST and FLASH LIST report image w x h for a slot holding an image, and RAM SAVE refuses one.

```
FLASH LOAD IMAGE 4, "tiles.bmp", 0        ' RAM slot 1, every start
TILEMAP CREATE mapdata, 1, 4, 16, 16, 16, 20, 15
BLIT FLASH 4, F, 0, 0, 100, 40, 32, 32, 0
```

- TILEMAP LOAD. TILEMAP LOAD file$, id, slot, tileW, tileH, tilesPerRow reads a tile map from a text file (.map by default), so a large map need not live in DATA.
- 32 DRAW3D objects. MAX3D is 32, up from 8, so a scene can hold 32 DRAW3D objects.

Fixed

Displays and modes

- PicoMiteVGA builds: the screen could come up garbled at power-on or reset, with bands of colour 12 pixels high either side of the middle of the screen in place of the sign-on, until a MODE command put it right. The firmware now waits for the video to start before it sets up the screen.
- PicoMiteHDMIBTH and PicoMiteHDMIWEB: OPTION RESOLUTION 800x600, 848x480 and 800x480 went back to 640x480 at the restart they cause.
- PicoMiteHDMI, PicoMiteHDMIUSB and PicoMitePC3, MODE 5 at 640x480 and 720x400: SAVE IMAGE saved each pixel's colour as its RGB332 value instead of the colour shown, so white was saved as 224, 224, 192 and a colour changed by MAP kept its old value. It, SAVE PNG and Print Screen now save the colours shown, MAP included, and PIXEL(x, y) returns the colour shown too, which changes what an existing program reads: white is now &HF8F8F8, where it was &HE0E0C0. A pixel copied with PIXEL x, y, PIXEL(a, b) is unchanged while MAP is at its defaults. As the manual now says, in MODE 5 MAP is used only at 640x480 and 720x400, and not on the PicoMiteHDMIBTH or PicoMiteHDMIWEB.
- MAP MAXIMITE striped every second column in HDMI MODE 2.
- MODE 1: text with a transparent background (TEXT with a background colour of -1) turned the background of its tiles white. BLIT of an area that lines up with the tiles left the tile colours behind when an area wider than one tile moved left or straight up or down, overwrote them when it moved down onto itself, and on HDMI at the resolutions with 8-bit tile colours never moved them.
- MODE 1: BLIT FLASH and RAY RENDER wrote 4 bits a pixel into MODE 1's 1-bit screen, up to three screens' worth past its end. They now stop with "Not available in mode 1".
- HDMI MODE 1 at 800x600, 848x480 and 800x480 (and at 640x480 and 720x400 on the PicoMiteHDMIBTH and PicoMiteHDMIWEB): FRAMEBUFFER LAYER and FRAMEBUFFER CREATE 2 were put over the tile colours, so the screen went black and drawing on them changed the colours. In MODE 1, FRAMEBUFFER LAYER TOP made a layer that no MODE 1 display shows; it now stops with "Invalid for this screen mode".
- VGA: TILE took an x one past the last tile across, which coloured the first tile of the next row, and a y one past the last row, which wrote past the tile colours.
- WebMite: with OPTION LCDPANEL VIRTUAL_C or VIRTUAL_M the first drawing command (even CLS) restarted the board.
- PicoMiteRP2350 with OPTION LCD SPI (which OPTION PLATFORM PICOCALC sets on that build): PIXEL(), BLIT READ and the other reads of an SPI panel came back with every colour bit one clock late, so each read and write cycle shifted the colours further. The reads now run at 12 MHz, or 6 MHz for the ILI9488 group, as on the other builds.
- PicoMiteRP2040MIN: OPTION LCDPANEL SSD1306I2C (and SSD1306I2C32) stopped with "SSD1963 displays not enabled", so an I2C OLED could not be set up.

Images

- SAVE IMAGE in MODE 1 on the HDMI builds lost the colours of the text: every colour but white, cyan and black came out black, so a picture of the colour-coded editor lost its numbers, strings and comments.
- SAVE IMAGE on a 16-colour display lost the last column of an odd width, and a 24-bit BMP's file size did not count the padding at the end of each row.
- A long SAVE IMAGE or Print Screen, which an SD card can stretch past a second, dropped a USB touch controller, and audio playback was not refilled until it ended. The save now services USB, audio, GPS and the keyboards between the rows of the picture, as LOAD BMP does, and the GUI cursor stays where it is until the save is finished.
- VGA and HDMI: a SAVE IMAGE stopped part way by a write error (a full or missing drive) left the layer (FRAMEBUFFER LAYER) in every later read of the screen, so PIXEL() returned the layer's colour wherever the

layer was not transparent, until a SAVE IMAGE completed. The error now clears it.
- LOAD IMAGE drew a top-down BMP (a negative height) upside down, and decoded a 16-bit BI_BITFIELDS BMP with a V4 or V5 header, as GIMP writes RGB565, with the wrong masks.
- A BMP file that could not be read left things behind: LOAD BMP leaked its buffers, the file stayed open (under ON ERROR SKIP or IGNORE each failure kept a file number), and SPRITE LOADBMP and BLIT LOAD left the buffer "in use". A file that is not a BMP is now "Invalid BMP file".
- FLASH LOAD IMAGE stored an odd-width image one byte short per row, so it drew skewed; it is now packed the way BLIT FLASH reads it. It now checks the whole file, and rejects a bad or oversize one, before it erases or writes the slot.
- LOAD JPG: an invalid ximage, yimage or scale was rejected only after the file had been opened, so each such error left a file number open and the eleventh gave "Too many files open". The values are now checked before the file is opened.
- RP2350: an interlaced or damaged PNG could crash LOAD PNG or SPRITE LOADPNG; they now stop with Interlaced PNG not supported or Invalid or damaged PNG file. LOAD PNG now refreshes a buffered display (one with OPTION REFRESH), as LOAD BMP and LOAD JPG do.

Sprites, framebuffers and tile maps

- SPRITE LOAD read one sprite more than its file declares (also from a blank line after the last one), refused a file whose last sprite would be the highest number, and loaded over an existing sprite of another size, overrunning its buffer, or over a copy or a sprite with copies; these are now refused with SPRITE READ's errors. It also accepted a width over 255, which overran its line buffer, and shifted every sprite after the first by a pixel when the width times the height is odd.
- SPRITE SWAP left both sprites using one background store, inside the old sprite's memory: showing the old sprite again overwrote the new one's background, and closing it freed the store still in use. It also left the old sprite's pixels where the new one is transparent, exchanged the two sprites' copy bookkeeping, so the original of a sprite with copies could be closed under them, and gave the new sprite the old one's bounds, which SPRITE CLOSE then freed twice.
- Sprite collisions after SPRITE MOVE, SCROLL and RESTORE: an overlap already reported was reported again and a new one could be missed, SPRITE(T, n) did not hold the sprites n overlapped, a layer-0 sprite met only layer-0 sprites, a sprite with a full list was left out of SPRITE(C, 0), and a second check before the collision interrupt ran could lose it. A sprite with three collisions that reached two screen edges wrote past its collision list, and its layer became -14.
- SPRITE WRITE of a sprite that is showing moved the sprite's recorded place and saved background to the copy, so HIDE removed the copy and left the sprite. SPRITE SHOW SAFE of a sprite that is showing ignored a new layer and the opaque orientations 4-7, and every redraw (MOVE, SCROLL, HIDE SAFE...) drew a sprite shown opaque as transparent again.
- SPRITE COPY accepted a count running past sprite 64 and lost count of copies 63 and 64, so their original could be closed.
- SPRITE(B, #n) read the wrong background pixels for a sprite of odd width and returned the previous function's value for a sprite not showing, and SPRITE(B, #n, side) failed for a sprite never loaded; a wrapping SPRITE SCROLL in MODE 1 overran its buffer. The sprite manual said a positive y scrolls down: it scrolls up.
- BLIT of compressed or packed data with an odd width to an LCD panel wrote one byte past a buffer on the stack. BLIT FRAMEBUFFER on VGA now refuses a source rectangle off the buffer.
- FRAMEBUFFER CLOSE L and CLOSE T were transposed and a selective close discarded the other buffers; the mode 5 layer and the mode 2 and 5 top layers were never freed.
- HDMI builds: FRAMEBUFFER LAYER drew behind the display in modes 2 and 5 when no LAYER TOP existed.
- RP2350 builds with PSRAM: FRAMEBUFFER LAYER (and LAYER TOP in MODE 3 and MODE 5) was refused with "Layer Buffer must be in tightly coupled RAM" whenever the layer was bigger than half the SRAM heap, even when the SRAM heap had the room: on the PicoMitePC3 a MODE 5 layer (76800 bytes) could not be made. A layer now takes the SRAM heap directly and is refused, with "Not enough RAM for the layer (n bytes)", only when the SRAM heap really cannot hold it. A 640x480 MODE 3 layer is 153600 bytes, more than the PC3's whole SRAM heap, so

that one is still refused, now with the reason.
- SAVE IMAGE, BLIT READ or PIXEL() after a framebuffer session hard-faulted, and the board rebooted, on an SPI panel that cannot be read back (ILI9488W, ST7789, ILI9481, GC9A01, the ST7735s and others). They now report "not available on this display", as they do before any framebuffer is created. On these panels SAVE IMAGE reads the framebuffer while FRAMEBUFFER WRITE F or L is in force and is refused for N; on panels with read-back it reads the screen for N.
- FRAMEBUFFER MERGE's background merge on an LCD panel overflowed core 1's stack on wide displays.
- PicoMiteRP2040MIN: RUN, NEW and an error did not close the program's tile maps, so after a program was run again TILEMAP could use, or free, memory that was no longer its own.

Drawing, fonts, TURTLE, DRAW3D and RAY

- TURTLE ARC LEFT and TURTLE ARC RIGHT stopped with "Expression syntax". ARCL, ARCLEFT, ARCR and ARCRIGHT always worked.
- LINE PLOT with a point count and a start or step in the array stopped early, or drew nothing: the array index was checked against the point count instead of the array's size.
- DRAW3D HIDE a, b left b showing when a was already hidden; a lit face (flag 8) was lit as the last face of the object was, whatever its own direction from the light ; DRAW3D CREATE accepted a face of more than 20 vertices, which overran the firmware's memory, and vertex numbers outside the object, which read outside it ; ROTATE, RESET, HIDE, RESTORE and CLOSE with no object did nothing instead of reporting "Argument count"; and DRAW3D CAMERA accepted a y below -32766.
- DRAW3D() never returned a value, DRAW3D CLOSE ALL left the last object, and LIGHT, SET FLAGS and RESET faulted on a missing object.
- RAY MAP with an integer array stored a cell of 32 or more as 32, one past the last wall type, so the wall's colours (and its door flag) were read from beyond the raycaster's memory. It is now 31, as the string form makes it. RAY RENDER left the row above the horizon from the last frame past a far wall, and three RAY messages showed 35c, 5d and 8d .
- A program could be left with the last program's fonts: after FLASH RUN, FLASH CHAIN, RAM RUN or RAM CHAIN, font 16 and any built-in font the last program had replaced with DefineFont were still there, and if that program's slot was then erased they became fonts with no size and FONT 16 gave no error. A program now starts with the built-in fonts, the library's and its own. If the selected font has gone the default font takes over, and a built-in font that comes back has its own size.
- A program that ends with its own font selected no longer breaks the next LOAD.
- Every built-in font now starts on a word boundary. Font 1 did not, in any build, so PEEK(WORD MM.INFO(FONT ADDRESS 1)) read the word two bytes before the font.
- Font 7's percent sign was mirrored.

Touch, GUI and the on-screen keyboard

- The on-screen keyboard stayed on the screen, and went on taking taps, in a program started by LOAD file$, R, CMM2 LOAD file$, R, FLASH RUN, RAM RUN, or F2 in the editor or the file manager: only RUN and CHAIN removed it, so whether a program had the full screen depended on how it was started. All of these now remove it, as the manual says, and a program shows it with KEYBOARD ON. In a program CHAIN, FLASH CHAIN and RAM CHAIN leave it as it was, displayed or hidden; CHAIN at the prompt removes it, as RUN does.
- KEYBOARD ON with no strip reserved said "Set OPTION SCREEN OFFSET first", an option that does not exist. It now says "Set OPTION SCREEN KEYBOARD first".
- GUI controls versions: if the GUI interrupt that a GUI CLICK set off ended in an error, the first GUI CLICK of the next run ignored its x and y and its interrupt was not called. If the pen-down interrupt for a TEXTBOX, NUMBERBOX, FORMATBOX or LISTBOX ended in an error, the next run's first return from any interrupt drew part of a pop-up keypad (in black) over its screen.
- A GUI cursor linked to the mouse left a copy of itself behind when the screen scrolled: at the console and the prompt, in the editor, in the on-screen keyboard's strip and with SPRITE SCROLL. The cursor is now taken off the screen before a scroll and drawn again straight after.

- The mouse cursor could leave a block of corruption behind on GUI and on-screen keyboard redraws; the on-screen keyboard's keys could not be clicked with a mouse under OPTION GUI CONTROLS 0; and the GUI TEXTBOX pop-up keyboard could not produce lower case.

Sound

The BBC Micro's SOUND and ENVELOPE statements arrive as PLAY BBC SOUND and PLAY BBC ENVELOPE on every board. Sound no longer stalls while pictures load, tones and sounds no longer slow down after other playback, and a directory's album may mix formats.

New and changed

- PLAY BBC SOUND and PLAY BBC ENVELOPE. PLAY BBC SOUND channel, amplitude, pitch, duration and PLAY BBC ENVELOPE are the BBC Micro's SOUND and ENVELOPE statements with the same parameters, units and behaviour, on every variant including the RP2040: three square-wave channels and a noise channel, sync marks for chords, &H10 to flush, sixteen envelopes with three-section pitch envelopes and ADSR amplitude at the authentic 100 Hz, and all eight of the BBC's noises, including the two clocked by channel 1's pitch. Each channel queues 16 notes, and MM.INFO(BBC BUFFER [channel]) reports how many more notes a channel will take without waiting; with no channel it gives the fullest of the four, the figure that is safe whichever channel comes next.

```
FOR p = 53 TO 101 STEP 4 : PLAY BBC SOUND 1, -15, p, 5 : NEXT p
```

- I2S audio uses the radio's PIO block on the wireless builds. I2S audio shares the radio's PIO block (PIO2, or PIO1 on the WebMiteRP2040), so on these builds that block is not available to your own PIO programs; on the WebMiteRP2350, PicoMiteHDMIWEB and PicoMiteHDMIBTH, PIO1 is now free for them. OPTION AUDIO I2S refuses pins the shared block cannot reach.
- Albums may mix formats. When PLAY WAV, FLAC, MP3 or MIDI plays a directory as an album, the directory may mix those four formats: PLAY NEXT, PLAY PREVIOUS and the start of the next track pick the decoder from each file's extension rather than from the one that is playing.

Fixed

- PLAY TONE and PLAY SOUND ran at half or a quarter of their speed and pitch after PLAY MODFILE, after a WAV, MP3 or FLAC file with a low sample rate, and, with PWM audio, after a USB device connected (its chime is an 8 kHz WAV): PLAY TONE 440, 440, 10000 played 110 Hz for 40 seconds. They now play at their own rate whatever played before.
- PLAY SOUND now reports an invalid type given as a string, and no longer takes the phase increment from the channel's old waveform when switching to or from white noise. On the VS1053 builds its right-channel sawtooth and a silent channel's DC offset are fixed. The computed square and sawtooth waveforms are now band-limited.
- A PLAY TONE that failed its checks cut a running endless tone to 1000 samples, and a finished sound's interrupt fired again when a later PLAY TONE, ARRAY or SAMPLE without one ended.
- PLAY ARRAY freed the buffers of whatever was playing instead of refusing with "Sound output in use".
- A play-once PLAY MODFILE's context was freed twice at its end, and its last buffer kept its earlier fill after the module ended, so a burst of earlier music followed the last note . PLAY MODSAMPLE with a volume of 0 played at full scale.
- VS1053 builds: PLAY HALT saved the position of its own .mem file instead of the track's for a track on A:.
- PLAY WAV with a file that is not 16-bit PCM (8-bit, 24-bit, floating point, A-law or mu-law) took 4 KB of free memory for every refill of its buffers, so with little memory left it stopped with Not enough memory for 4096 bytes, or the sound stopped part way through the file with no error. It now takes 512 bytes, with no audible difference.
- PLAY WAV, FLAC, MP3 or MIDI on a directory found it only when its drive was the current one, and otherwise reported "Could not find the file". The drive is now taken from the name, so PLAY FLAC "B:/music" works with A: as the current drive.
- Sound played with PLAY (WAV, FLAC, MP3, MIDI, MOD, tones, sounds and arrays, and on the Bluetooth builds a speaker) stalled while LOAD BMP, LOAD IMAGE, LOAD JPG or LOAD PNG decoded a picture, and while SPRITE

LOADBMP, BLIT LOADBMP or SPRITE LOADPNG read one. The decoders now keep the sound fed as they go, on every build. FLASH LOAD IMAGE into a flash slot still stops the sound: the flash is being written.
- The USB connect and disconnect chime, when it finished, closed the file number of the last WAV file played, which could be a file opened since, and before any WAV file had played it could raise an old file error.
- Wireless builds: a clock change (CPU SPEED, or RESOLUTION on the HDMI builds) retuned every state machine on the radio's PIO block, so I2S audio on that block could then play at the wrong speed. Only the radio's state machine is retuned now.
- PicoMiteRP2350BT, PicoMiteRP2350BTH and PicoMiteHDMIBTH: I2S on GP32 or above could not work, because the radio had already fixed the PIO block's range of pins.
- OPTION AUDIO: with OPTION AUDIO I2S set, the PWM, SPI and VS1053 forms were accepted too, leaving their pins reserved but silent and OPTION LIST showing only one output. They now give Audio already configured.

Files and drives

Files and drives gain ZMODEM, a transfer that recovers from lost data and can move many files at once, and the A: drive moves to a newer littlefs that copes better with power failures. Faults with SD cards, A: directories, SEEK, TFTP and programs written to flash are fixed.

New and changed

- ZMODEM file transfer (every build). ZMODEM moves a program or a file between the board and your computer over the console, the USB console or a serial one, in the same forms as XMODEM and YMODEM. The file forms also work inside a program; the program forms are for the command prompt. SEND, RECEIVE and CRUNCH can be shortened to S, R and C. A program too large for the receive buffer stops with Not enough memory, and the program in memory is left as it was.

```
ZMODEM SEND                  ' the program in memory
ZMODEM SEND "B:/data.csv"    ' a file on A:, B: or C:
ZMODEM RECEIVE               ' a program into program memory
ZMODEM RECEIVE "B:/data.csv" ' a file
ZMODEM RECEIVE "B:/dl/*"     ' any number of files, under the sender's names
ZMODEM CRUNCH                ' a program, without comments, blank lines and spaces
```

- ZMODEM recovers instead of failing, and is fast. Every block carries a CRC-32 and its place in the file; when a block is lost or damaged the receiver asks for the data again from that point, so a poor connection slows a transfer down rather than stopping it. Over the USB console with Tera Term a 512 KB file takes about 2 seconds on an RP2350 and 3 seconds on an RP2040. Over a serial console it depends on the baud rate: through a PicoComputer 3's CH340 at 750000 baud, about 70 KB a second. Some adapters, the CH340 among them, lose data on the way to the computer at 921600 baud; ZMODEM recovers, but 750000 is faster.
- ZMODEM at the computer's end: Tera Term on Windows (File > Transfer > ZMODEM > Send or Receive), or sz and rz from the lrzsz package on Linux and macOS. The board waits 60 seconds for the other end to start, and a transfer cancelled there stops with Cancelled by remote.
- ZMODEM names and batches of files. ZMODEM SEND gives the computer the name of the file the program was loaded from, or FILE.DAT; ZMODEM SEND file$ gives the file's name without its drive or directory. ZMODEM RECEIVE file$ saves under file$ whatever name the sender gives, and of a batch keeps only the first. ZMODEM RECEIVE "*" takes every file the sender offers and saves each one in the current directory under the sender's name (without any directory it sends), overwriting a file of that name; put a directory in front of the * to receive into it, as in ZMODEM RECEIVE "B:/dl/*", and select the files in Tera Term's ZMODEM Send dialog or list them to lrzsz (sz *.bas). It ends with Received 3 files, 221000 bytes; if an error stops it part way, the files before it are complete and the error says how many there were, as in The path name format is invalid (files received: 1).
- The A: drive uses littlefs 2.11.3 (it was 2.5.1), with three years of littlefs fixes. A write cut short by a power failure is now caught by a checksum, where before it could be read back as good data; on a drive more than half full, the two blocks that describe the drive were rewritten so often that they could wear out first; tidying up a file or directory "orphaned" by a power failure could loop for ever; and SEEK to near the end of a 4 KB block, or a file open twice

and written through one number and read through the other, could read old or corrupted data. Names still ignore case. An A: drive written by V6.03.01 is kept, but the first write to it converts it to littlefs's newer format, which earlier firmware cannot read: going back to V6.03.01 reformats the A: drive, so copy anything you need first.
- MM.INFO$(USBDRIVE) gives the state of the USB drive C: as MM.INFO$(SDCARD) gives B:'s: "Ready" when a USB drive is plugged in and mounts, "Not present" when none is, and "Not Configured" on the builds that have no C: drive. Like SDCARD it never stops with an error, so a program can test it before it uses C:.
- FLASH ERASE works from inside a running program, so a program can manage its own slots.

Fixed

Programs and the library

- CMM2 RUN and CMM2 LOAD changed the current directory to the program's directory and only then opened the program by the name they were given, so CMM2 RUN "tmp/foo.bas" from the directory above looked for tmp/tmp/foo.bas ("Could not find the path"), a program on another drive stopped with "Only valid on current drive", and one that worked left you in the program's directory. The current directory is now never changed, as on the Colour Maximite 2, and #INCLUDE still finds a file beside the program wherever it is run from.
- #INCLUDE "/lib/x.inc", a path from the root with no drive, was taken as relative to the program's directory. It is now from the root of the current drive, as on the Colour Maximite 2.
- LOAD (and the RAM program loader) reported a program line longer than 255 characters as "Not enough memory". It now says "Line is too long".
- An error while a program was being written to flash left the board unresponsive part way through printing the message, while the display kept running. The message is now printed and the prompt returns.
- RAM FILE LOAD left the running program without its subroutines and wrote 1 KB past the PSRAM allocation map, and RAM ERASE ALL cleared 128 KB beyond the end of the PSRAM and into the heap. Both are fixed.
- A corrupt library could lock a board out: a damaged image hung the interpreter before the prompt, and reflashing did not help because a .uf2 leaves the library where it is. The library is now checked at boot, and one that fails is erased with the message Library was corrupt and has been discarded.

A: drive

- OPEN "dir" FOR OUTPUT on a directory on A: gave no error, and only the first write failed. It is now refused with "Entry is a dir".
- A DIR$() loop that KILLs or renames the files it finds on A: missed some of them, because DIR$() carried on reading a directory it had already closed. The directory now stays open, and up to date, until DIR$() returns an empty string.
- Ctrl-C at KILL's (Y/N) question, or an error part way through FILES, or through KILL or COPY with a wildcard, left an A: directory open, and the next of these commands, or CHDIR, could then hang the board at its next write to A:. A directory left open this way is now closed before it is used again.
- On a board whose 3.3 V rail sits close to the flash chip's minimum, the boot counter written to A: early in start-up could reach the wrong sector, leaving a board that booted once more and then not at all. The first flash write after a power-on now waits a second for the supply to settle; a reset button or CPU RESTART does not wait.
- A drive format requested during start-up is no longer repeated by an unrelated OPTION PLATFORM change afterwards.
- Writes to the A: drive are now bounded at run time, so a bad block number can no longer become an erase at some other address in the flash.

SD card and USB drive

- Mounting the SD card closed every open file, the A: drive's included: a file opened on A: before the first access to B: (or the first after an SD card error) was gone, and the next SEEK or LINE INPUT on it gave "File number". Mounting B:, or the USB drive C:, now closes only the files open on that drive.
- A disk error on a USB drive (C:) was taken for one on the SD card (B:): the SD card was marked to be mounted again, which broke a file open on B:, while C: itself was left as it was. C: errors are now treated as C:'s.

- A file operation straight after a background FRAMEBUFFER COPY ,B (or MERGE ,B / BLIT ,B), such as PLAY MODFILE, LOAD BMP or a file read, failed with an I/O error when the SD card shares the system SPI with the display; the only cure was a PAUSE tuned to the CPU speed. File access to the SD card now waits for the copy to finish. The continuous background MERGE is unchanged: a program still has to keep file access away from it.
- After SEEK on an SD card, reading part of a file (MEMORY INPUT, LONGSTRING LOAD, structure LOAD) started up to 512 bytes late, and LOC() could be wrong. EOF() could say there was more after a seek to or past the end, on SD and on flash.

TFTP (WebMites)

- On the WebMites (WebMiteRP2040, WebMiteRP2350, PicoMiteHDMIWEB), a TFTP request for B:/name was read from or written to the current drive instead of B: after an error, after leaving EDIT or an XMODEM transfer, or after a command with no arguments such as CLS. The drive in the request is now always the one used.
- A TFTP transfer that failed, such as a request for a missing file or a missing directory, left the computer waiting until it timed out, set MM.ERRNO, and stopped a running program with an error it had not caused (Could not find the file); the next transfer was then refused. The computer now gets a TFTP error at once, the console shows TFTP error: and the reason, and a running program carries on.

USB devices

The USB host builds have an up-to-date USB stack: several devices behind a hub come up reliably, a USB flash drive reads several hundred KB/s, more serial adapters, gamepads, mice and touch screens work fully, and recoverable USB faults no longer take the interpreter down.

New and changed

- An up-to-date USB host stack. The USB host builds use TinyUSB's current development version, with PicoMite's patches on top. When a keyboard, mouse, game controller or flash drive behind a hub does not answer while it is being set up, it is now reset and set up again, so several devices behind a hub come up reliably on every kind of reset and power-on. The connect chime plays once, after the device is set up. Every other build uses the same TinyUSB for its USB console.
- Faster USB flash drive (C:). Reads run at several hundred KB/s instead of about 64 KB/s, faster than the SD card on the same board, so PLAY MP3 "C:/track.mp3" streams without underruns.
- More USB serial adapters on COM3 to COM6. An adapter with a Prolific PL2303 chip, which was ignored, now opens like FTDI, CP210x and CH34x adapters. A device that refuses DTR and RTS when it is plugged in, such as the Microbridge (the USB serial chip on a Colour Maximite 2 Generation 1 and other boards), was never connected, and on a PicoComputer 3 no other device on the hub was connected after it; it now connects, and OPEN opens it without DTR and RTS, at up to 115200 baud. A board with an MCP2221A bridge works only at up to 57600 baud: it has no flow control, and at 115200 it loses characters on continuous output.
- A USB serial adapter receives only while its port is OPEN. What arrives while the port is closed is dropped, as on other ports, and an unopened adapter no longer takes turns on the bus the USB drive uses.
- GPS on a USB serial port: OPEN "COM3:" AS GPS through COM6: now starts the GPS parser, as the UARTs always did.
- More gamepads. The Nintendo Switch Pro Controller, and the gamepads that present themselves as one in their Switch mode, are supported; before, it connected and dropped off the bus every few seconds. Its buttons, d-pad (also as DEVICE(GAMEPAD n, H)), sticks and triggers map onto DEVICE(GAMEPAD n, ...) as the other pads' do, and MM.INFO(USB n) reports it as 135. It leaves the bus while the firmware writes its flash (LOAD, SAVE, RUN "file", ...) and reconnects a moment later. A generic gamepad with VID &H0810 and PID &H0001 is now one of the built-in pads, with the same buttons as the other SNES-style pads (DEVICE(GAMEPAD n, B)), and no longer needs GAMEPAD CONFIGURE.
- Every mouse button. With OPTION MOUSE SENSITIVITY set, DEVICE(MOUSE n, B) and the mouse interrupt now see every button a mouse declares, not only left, right and middle: bits 0 to 2 are left, right and middle, and the higher bits the mouse's further buttons in its own order (a five-button mouse puts its side buttons in bits 3 and 4). In

boot mode, without that option, a mouse reports three buttons.
- Room for 31 USB endpoints, up from 15, which a hub, a serial adapter, a keyboard with media keys, a mouse and a flash drive used up. A device that still does not fit is reported as [USB EP_FULL ...].
- USB faults recover and report. Recoverable USB conditions are recovered from and reported ([USB DATA_SEQ 3 @181]) instead of taking the interpreter down with a bare *** FAULT. The counts are readable with PEEK. Bit-stuff, CRC and receive-overflow errors on the bus are counted and reported with them; the first of each kind prints a line.

Fixed

- A hub that switched its ports off after an over-current left them off, and a port that the hub disabled by itself stayed dead with its device still plugged in. The ports are now powered again once the over-current is over, and a device on a disabled port is enumerated again.
- A key held down as its USB keyboard was unplugged repeated for ever; the repeat now stops.
- Every build whose console is the USB port (all but the USB host builds and the PicoMiteRP2350BT): the console could lock up for good, with USB and the heartbeat stopped, until a reset or a power cycle. It took characters arriving one per USB packet, as when a PicoMite USB host prints to another PicoMite on its COM3 or a PC program writes byte by byte. This is fixed.
- OPEN "COM3:..." to a USB serial device that refused the baud rate or the DTR/RTS setting still opened, and then nothing went either way, with no error at either end. It now stops with COM3: USB device did not accept the baud rate (or ... DTR/RTS); only a device that refuses DTR and RTS from the moment it is plugged in, such as the Microbridge, is opened without them. A PicoMite at the other end only reads and writes its USB console while DTR is set.
- On a USB COM port, CLOSE dropped DTR with data still unsent, FLUSH did not wait for it and LOF did not count it. CLOSE now keeps DTR until the data has gone, FLUSH waits for it and LOF counts it .
- A keyboard with media keys can present two boot-keyboard interfaces, and the second took a HID slot of its own: the console showed "USB Keyboard Connected on channel 1" and again "on channel 4", and a gamepad or second mouse had one slot fewer. The second interface now takes no slot.
- Gamepads: a DualShock 4's d-pad gave down for down-left and left with down for up-left. An unplugged gamepad kept its last buttons and sticks in DEVICE(GAMEPAD n, ...); they now read 0 .
- GAMEPAD CONFIGURE's codes 128-135 tested the wrong bit and 128 was refused, and a code it could not use raised an error from inside the USB stack; such a code is now refused when it is given. GAMEPAD MONITOR OFF now ends the monitoring, which nothing could before, and a generic pad plugged in while a program runs no longer prints a blank line.
- Touch monitors with the WingCoolTouch/WCH controller (USB ID 27C0:0859, also in the Corsair Xeneon Edge and the Verbatim HD14T) connected as "Multi-touch" but TOUCH() never saw a touch; they now work. On a panel that reports a newly landed finger first, such as this one, TOUCH(X) and TOUCH(Y) jumped to a second finger while the first was still held, TOUCH(X2) and TOUCH(Y2) gave the first, and an on-screen keyboard chord (Sh or Ctl held, then a key) did nothing; each finger now keeps its place while it stays down.
- USB touch screens whose Input Mode report carries a report ID, such as a SiS panel (0457:0819), could give only one contact, because the switch to touch screen mode wrote the mode over the ID. They are now switched to touch screen mode as Windows does it.
- On a USB touch screen without OPTION GUI CONTROLS, TOUCH(TAP), TOUCH(DTAP), TOUCH(HOLD) and the swipes never fired; only the pinch and rotate gestures did. They all work now, with or without GUI controls.
- MM.INFO(USB n) gave 25 for a multi-touch screen instead of 133, the code the manual gives, so a program looking for 133 could not find it. It now gives 133.
- MM.INFO(USB VID n) and MM.INFO(USB PID n) gave 0 for a touch screen on channel 4, and gave a device's IDs for an empty channel: the channel first chosen for a touch screen, or one left by a device with an interface the firmware does not use, such as a touch monitor's vendor interface. Channel 4 now gives the touch screen's IDs, and an empty channel gives 0.

Pins, serial ports and devices

COM ports now finish sending before a program ends, the clock's milliseconds behave as documented and can follow a time server or a GPS pulse, and I2C, SPI and ONEWIRE READ take structure members. Many faults in serial ports, GPS, pin set-up and interrupts are fixed.

New and changed

- A program's end sends what its COM ports hold. A program that ended used to close its COM ports at once and lose whatever they still had to send, up to 256 bytes on COM1 and COM2: PRINT #5, ... at 300 baud followed by END sent nothing. Now, however it ends (END, its last line, an error or CHAIN, and including what MM.END sends), the ports send what they hold before they close, as on the Micromite. Ctrl-C still ends a program at once and drops what is unsent, and a Ctrl-C during that wait, or while CLOSE or FLUSH waits, stops it at once, from any console.
- Milliseconds, WEB NTP and the RTC. With OPTION MILLISECONDS ON the milliseconds of TIME$ now start at 0 when the time is set (TIME$, RTC GETTIME), as documented, and WEB NTP sets them from the time server, to within a few milliseconds. TIME$ = n, which moves the clock by n seconds (now in the manual), also sets them to zero, so TIME$ = 0 at a pulse from an external source, such as a GPS receiver's one pulse per second output, lines the clock's seconds up with it without changing the time. OPTION RTC AUTO ENABLE reads the RTC at boot and then once a day, not every hour, and WEB NTP restarts the day, so the RTC's whole-second time no longer overwrites a time set from the network.
- I2C READ, SPI READ and ONEWIRE READ into a structure member (and their second-channel and slave forms), as on the CMM2: a string member, up to its own LENGTH, or a number, alone or in a list (SPI READ 2, r.v, r.f). A whole structure, or a member of an element of a structure array given alone, is still refused. Not on the PicoMiteRP2040MIN, which has no structures.

Fixed

Serial ports

- CLOSE and FLUSH on COM1 and COM2 could cut off the last one or two characters: they waited for the transmit buffer to empty, but not for the UART to send what it still held. They now wait for the last character to go.
- On a serial console (OPTION SERIAL CONSOLE) and on COM1 and COM2, the last character of some output could stay unsent until the next character was sent. Every character now starts the transmitter once it is queued.
- Ctrl-C while output was waiting for room in a serial port's transmit buffer (COM1, COM2, a USB serial port or Bluetooth COM7), for example a long PRINT #n at a low baud rate, went straight back to the prompt without tidying up: the files stayed open and the last command line came back at the prompt. It now ends the way any Ctrl-C does.
- SETPIN a, b, COM1 or COM2, refused because the console or GPS uses that UART or because the port already has other pins, still changed the pin it refused: the pin became a plain input with no pull-up or pull-down, but kept its old setting in the pin list. A refused SETPIN now leaves the pin as it was.
- OPTION SERIAL CONSOLE and OPTION GPS accepted two TX pins, or two RX pins, of the same UART, and saved them with whatever the other pin had been, so the console or GPS could restart with a pin from its previous setting or with no pin at all. They now need one TX pin and one RX pin, and give "Invalid configuration" otherwise.

GPS and time

- While the receiver has no fix it sends its sentences with the values left empty. GPS(GEOID), GPS(ALTITUDE), GPS(DOP), GPS(SATELLITES), GPS(FIX), GPS(SPEED) and GPS(TRACK) then kept the values of the last fix, while GPS(LATITUDE) and GPS(LONGITUDE) went to 0; they now read 0 too. A sentence with fewer fields than expected no longer makes the firmware read past the end of it.
- Until the GPS sent a sentence with the date, GPS(TIME) and GPS(DATE) returned 48 characters of whatever was in memory. They now return 00:00:00 and 00-00-2000.
- PRINT #GPS included the $ in the NMEA checksum it appends and sent a checksum digit from A to F as K to P, so commands reached the receiver with a wrong checksum and were ignored. It also refused with "GPS not activated" when the receiver was set up with OPTION GPS; it now writes to that UART.
- CLOSE g, OPEN ... AS g and PRINT #g took any name starting with G as GPS.

- STAR with a GPS opened with a time offset (OPEN ... AS GPS, offset) used the local time as UT, putting the star out by up to 15 degrees per hour of offset. It now takes the UT time of the last GPS sentence.
- The GPS monitor could overflow the stack when printing to a display console.
- With OPTION RTC AUTO ENABLE, the clock read at the command prompt took a temporary string that was never freed, so after about 64 hours idle at the prompt the board reported "Not enough memory".

I2C, SPI, pins and interrupts

- SPI READ and SPI2 READ with no destination, as in SPI READ 3, used an argument that was never set, giving an odd error or worse. It is now "Invalid syntax".
- I2C READ, SPI READ and ONEWIRE READ (and their second-channel and slave forms) into a string checked the count only against 255, not against the string's LENGTH, so a longer read wrote past the string and could corrupt that variable and the ones after it. It is now "Number out of bounds".
- PORT(...) = value and PORT(...) with more than 31 pins (RP2350B) repeated bit 31 into the higher pins, or lost the bits above 32.
- WS2812 with an array smaller than the LED count read past the array and sent what followed it to the LEDs; it is refused with "Array too small".
- PicoMiteHDMIWEB, PicoMiteHDMIBTH and PicoMitePC3: ADC RUN, and ADC START with an interrupt, shared a DMA channel with the radio. The radio's first transfer stopped the capture, and the ADC interrupt then fired on the radio's traffic instead of on full buffers. This is fixed.
- ON KEY k now works on a UART serial console; SETTICK refuses a period of -1; ONESHOT no longer shares its callback bit with the PS/2 mouse; an I2C slave interrupt no longer loses a ready bit; and a PIO TX interrupt uses each state machine's own FIFO depth.
- PIO SYNC wrote the wrong PIO's CTRL register, and PIO DMA completion disabled the wrong state machine on PIO 2.

Stepper motors (RP2350)

STEPPER, already in V6.03.01, now blends arcs into a continuous path, works alongside I2S audio and is stopped when a new program starts. Faults in corner speeds, step counts, pulse timing, pin set-up and E-STOP handling are fixed.

New and changed

- Arcs blend with the moves around them. In G64 (continuous) mode G2 and G3 now blend with the moves before and after them, as G0 and G1 already did, instead of stopping at each end. A path built from short arcs and lines, as CAM programs write them, no longer slows towards a stop at every move: at 500 mm/s^2, a 20 mm circle at F1200 drawn as 36 short arcs now runs at 98% of the feedrate (71% before), and drawn as 36 lines at 97% (74%); at F3000 the arcs reach 73% (31%) and the lines 80% (30%). G61 still stops at the end of every move.
- RUN, NEW, LOAD and CHAIN stop the stepper, as do FILES, EDIT FILE and RAM FILE LOAD run from inside a program. It used to keep running over the old program's memory, so a queued job could execute garbage, and the next program could not STEPPER INIT without a STEPPER CLOSE. A program now simply starts with STEPPER INIT.
- The stepper works with I2S audio (OPTION AUDIO I2S); STEPPER commands used to report "Stepper incompatible with I2S audio". Audio keeps playing while the motors run, though the sound can change briefly while STEPPER GCODE commands are being queued; the motion is not affected. STEPPER INIT still refuses to start if the audio uses PWM slice 10.

Fixed

- The corner speed in G64 mode was worked out the wrong way round: a gentle turn was taken at a crawl (about 1.5 mm/s for a 10 degree turn at 500 mm/s^2) and a near-reversal at almost full speed, which can lose steps. Gentle turns are now taken fast and sharp ones slowly, and no corner is taken faster than the following move can stop from.

- G0 and G1 rounded each move's distance down to whole steps, so every move could lose a step on each axis and a long program of short moves drifted: ten moves of 0.4358 mm ended 9 steps short. Moves now end on the nearest step to their target, as arcs already did.
- In an arc the direction output could change while a step pulse was still high. It now changes only after the pulse has ended.
- G28 homing gave 2 us step pulses, 2 us after setting the direction, against 10 us in normal motion; some drivers, servo drives in particular, miss pulses that short. Both are now 10 us, at the same homing speed.
- The limit switch inputs (STEPPER HWLIMITS) and the E-STOP input (STEPPER INIT) lost their pull-up (pull-down for an inverted limit) as soon as they were set up, so a switch that was open, or not wired, left the input floating and could trip at random. The pulls now stay on.
- STEPPER AXIS left the enable pin low, which turns most drivers on, and STEPPER SPINDLE pin, 1 (active-low) left the spindle running. The enable pin now starts high (drivers off) until STEPPER ENABLE, and the spindle off until M03.
- After a hardware E-STOP trip in a program, the next STEPPER command printed the trip report and then did nothing, without an error: a STEPPER POSITION set nothing, and a STEPPER CLOSE left the pins held, so the next STEPPER INIT gave "Stepper already initialized". The command now runs after the report.
- STEPPER ESTOP during an arc no longer leaks the arc's buffer.

Known limitation

- Arcs run faster than their feed rate. A G2 or G3 arc holds whichever axis is moving faster to the feed rate, rather than the speed along the arc. Where the arc runs along an axis it goes at the feed rate; elsewhere it goes faster, up to 1.41 times the feed rate at 45 degrees and about 1.11 times on average round a circle: a semicircle at F3000 (50 mm/s) reaches 70.6 mm/s at its 45 degree points. No axis goes over its STEPPER AXIS maximum speed, but a cutter, laser or pen moves along an arc faster than programmed; straight moves (G0, G1) are not affected, and the arc figures above include this overspeed. It will be corrected in a later release (issue #161).

Boards and builds

There are two new builds: the PicoMitePC3, one firmware with HDMI, USB host, WiFi and Bluetooth for the Pico Computer 3 that also runs the Pico Computer 2 and Olimex's RP2350pc, and the PicoMiteHDMIBTH, an HDMI and USB host build with Bluetooth. The RP2350 wireless builds can use a radio wired to other pins, the flash size is now measured, and several PSRAM faults on the RP2350 are fixed.

New and changed

The PicoMitePC3

- The PicoMitePC3: everything the Pico Computer 3 can do, in one firmware. It combines the PicoMiteHDMIUSB's HDMI video (the full set of resolutions and modes, USB keyboards, mice and game controllers, the OV7670 camera), the PicoMiteHDMIWEB's WiFi (TCP, UDP, TLS, MQTT, NTP, TFTP, the web server) and the PicoMiteHDMIBTH's Bluetooth (keyboards, mice and game controllers, speakers and headphones, serial devices on COM7). It needs PSRAM, and runs on the Pico Computer 3, the Pico Computer 2 (the Pico Computer 3 without its WiFi and Bluetooth module) and Olimex's RP2350pc. MM.DEVICE$ reads PicoMitePC3 RP2350B and the sign-on is PicoMitePC3. Its first boot is the full reset described under Upgrading from V6.03.01.
- 256K of program memory, against 180K on the PicoMiteHDMIBTH and 156K on the PicoMiteHDMIWEB. The editor, LOAD, AUTOSAVE, LIBRARY and the XModem, YModem and ZModem program transfers take their buffers from the PSRAM, so a 250K program can be edited and saved. With the PSRAM disabled (OPTION PSRAM PIN DISABLE) only a program that fits in the internal memory can be loaded. The PSRAM reserve is 2.5 MB rather than 2 MB, so MM.INFO(PSRAM SIZE) reads 5767168 on an 8 MB part.
- It sets itself up for the board it is on. At a boot with no platform set (a new board, after OPTION RESET on its own, or once the options have been reset) it looks, in turn, for the 32 kHz clock from the Pico Computer 3's real-time clock on GP27 (OPTION RESET PICO COMPUTER 3), for PSRAM on GP47 (the Pico Computer 2: OPTION RESET

HDMIUSBI2S, which on the PicoMitePC3 also sets the PSRAM on GP47 and turns the radio off) and for PSRAM on GP8 (the RP2350pc: OPTION RESET RP2350PC). The board restarts once with that configuration. After that the platform is set and the boot looks no further, so the options you change stay as you set them. A board that matches none of the three keeps the default options.

- OPTION RESET RP2350PC puts the PSRAM (8 MB) on GP8, the HDMI output in the default pin order, the SD card on GP9 (CS), GP10, GP11 and GP24, stereo audio as PWM on GP26 and GP27, the serial console on GP0 and GP1 (UART0, on UEXT1) and the heartbeat on the LED at GP25, with no radio. Four more pins are the board's and are reserved at boot: GP32 and GP33, the audio codec's I2C, are pulled high so the codec stays idle (leaving I2C0 free for UEXT2's I2C on GP36 and GP37); GP35 is held low to keep the amplifier on; and GP34 sets the amplifier's volume with PWM channel 9A, at 50%. PWM channel 9 is therefore not available to programs ("Channel in use for amplifier"), and LIST PINS names the four pins. Use the PicoMitePC3 firmware on this board: the PicoMiteHDMIUSB starts its serial console on GP8, which on the RP2350pc is the PSRAM's chip select, and stops at the first character.
- OPTION RADIO BOTH | WEB | BLUETOOTH | NONE chooses which halves of the Pico Computer 3's wireless chip the next boot brings up, so the one build serves as the all-in machine, a WebMite, a Bluetooth host or a plain HDMI computer, with the half not in use costing nothing. BOTH is the default and lists nothing in OPTION LIST. WEB leaves the Bluetooth host out; the BLUETOOTH commands and OPTION AUDIO BLUETOOTH answer "Bluetooth is off (OPTION RADIO)". BLUETOOTH leaves the WiFi down (no join, no TCP, TFTP or UDP servers, no NTP); the WEB commands answer "WiFi is off (OPTION RADIO)". NONE leaves the chip unpowered and untouched: its four pins are ordinary I/O, and the heartbeat LED is on GP25, as on the PicoMiteHDMIUSB, where OPTION HEARTBEAT OFF and ON act on it. The setting is saved and the firmware restarts. The stored network and the Bluetooth pairings are kept, so any setting can be changed back; the Bluetooth host cannot be switched off while the audio is on Bluetooth.
- Speed. Measured at 315 MHz, solar_eclipse runs in 14.15 s with OPTION RADIO NONE (the PicoMiteHDMIUSB's speed), 14.57 s with WEB and WiFi joined, 15.11 s with BLUETOOTH and 15.04 s with BOTH. With Bluetooth on it is as fast as the PicoMiteHDMIBTH (within 2%) and 7 to 9% behind the PicoMiteHDMIWEB; the difference is the Bluetooth host's polling, which every Bluetooth host build pays. Running the heap in PSRAM costs about 1%.
- WiFi and Bluetooth share one radio. While a Bluetooth speaker is playing, the radio gives the sound priority: a WiFi transfer slows to a crawl until the sound stops and recovers the moment it does. A speaker that is connected but silent costs WiFi nothing. With Bluetooth on, WEB SCAN has the radio to itself for the few seconds it takes, so it finds and reads networks as it does with Bluetooth off; meanwhile a speaker's sound stops and other Bluetooth links wait.
- Sound, keyboards and transfers. Sound goes through the Pico Computer's I2S DAC, a Bluetooth speaker, PWM (OPTION AUDIO pin, pin) or an SPI DAC (OPTION AUDIO SPI); there is no VS1053 audio. As on the PicoMiteHDMIBTH there is no PS/2 or I2C keyboard. There is no YModem on the USB console (the console is a serial port on the PC3); XModem and ZModem work as on every build.

Other builds and boards

- A new build: PicoMiteHDMIBTH. This is the PicoMiteHDMIUSB (HDMI display and USB host) with the PicoMiteRP2350BTH's Bluetooth added: keyboards and mice, speakers and headphones, and serial devices on COM7 (see the Bluetooth chapter). Like the PicoMiteHDMIWEB it needs an RP2350 board with the Raspberry Pi wireless chip (a Pico 2 W or similar). It takes programs of up to 180 KB.
- OPTION CYW43 PINS wl_on, wl_d, wl_cs, wl_clk: modules that wire the wireless chip differently. The WiFi and Bluetooth builds expect the radio on the Pico 2 W's pins, GP23 (power on), GP24 (data), GP25 (chip select) and GP29 (clock); this option sets others. Waveshare's RP2350B Pico WiFi, for example, needs OPTION CYW43 PINS GP36, GP37, GP38, GP39. The pins given are reserved, and GP23, GP24, GP25 and GP29 become normal I/O pins. OPTION CYW43 PINS DEFAULT, or OPTION RESET, goes back to the standard pins. An I2S DAC shares a PIO block with the radio, so its pins and the radio's data and clock pins must all be within GP0 to GP31 or all within GP16 to GP47. If the radio does not answer on the pins set, the board says CYW43 radio failed to start on GP36,GP37,GP38,GP39 (OPTION CYW43 PINS) (with the pins set) as it starts and carries on without WiFi or Bluetooth; the WEB commands then report WiFi radio not running. This is in the five RP2350 wireless builds: the

WebMiteRP2350, PicoMiteRP2350BT, PicoMiteRP2350BTH, PicoMiteHDMIWEB and PicoMiteHDMIBTH. The WebMiteRP2040's pins stay fixed.
- The flash size is measured. OPTION FLASH SIZE (MM.INFO(OPTION FLASH SIZE)) was the size the flash chip reports in its ID, which not every maker encodes the same way. That figure is now checked by reading past the end of the chip, as the PSRAM's size already was, and the size is measured at every boot.
- PicoCalc: KEYDOWN() works on the built-in keyboard.

Fixed

- OPTION RESET LIST on the PicoMiteHDMIWEB listed the Palm Pico and USB Edition V1.0 presets, which are for LCD boards, and on the WebMiteRP2350 the Palm Pico, which it does not have. Presets are now listed only where they exist.
- WebMiteRP2350, PicoMiteRP2350BT and PicoMiteRP2350BTH: OPTION PICO PLUS 2 ON did not work on the Pimoroni Pico Plus 2 W, because it tried to put the heartbeat on GP25, which on that board is the wireless chip's select line. It now enables the PSRAM on GP47 as on the Pico Plus 2 and uses the LED on the wireless chip as the heartbeat. OFF leaves that heartbeat on.
- PicoCalc: OPTION PLATFORM "PICOCALC" written in another case than "PicoCalc" turned off its options. Ctrl+C typed quickly no longer produces a literal c.
- RP2350 with PSRAM (a Pico Computer 3, for example): the board could stop with a hard fault, usually at the command prompt, after a program used WS2812 or BITSTREAM, or after anything that writes flash (saving a program, writing a file on A:, changing an option). It no longer does.
- RP2350 with PSRAM: writing or closing a file on A: could throw away data a program had just stored in PSRAM, so a large array or string could lose its most recent changes.
- RP2350 with PSRAM: CPU SPEED, and on the HDMI builds a RESOLUTION that changes the clock, left the PSRAM timed for the clock the board started at. A higher clock ran the PSRAM past its rating (started at 200 MHz, then CPU SPEED 320000: 160 MHz, for a part rated 104 or 109 MHz), and a lower one held it selected for longer than the part allows between refreshes. The PSRAM timing now follows the clock.
- RP2350: a PSRAM part larger than 8 MB made the memory allocator write past its page map on its first allocation. Such a part is now used as 8 MB, with a line at boot saying so; the rest is left unused.
- RP2350: a PSRAM that did not answer at boot put the board into a silent boot loop that also rewrote the option sector on every cycle. The boot now carries on, reports PSRAM not responding once the console is up and keeps the option for the next boot.

WiFi (WebMite builds)

The WiFi builds (WebMiteRP2040, WebMiteRP2350, PicoMiteHDMIWEB and the new PicoMitePC3) gain MM.INFO(TELNET), run a little faster and take the TLS handshake's memory from the PSRAM where there is one. Most of the work is fixes: to the web server and Telnet, WEB TCP CLIENT, HTTPS downloads, the TCP and UDP interrupts and JSON$.

New and changed

- RUN closes an open TCP client connection, as it already did MQTT.
- MM.INFO(TELNET) is 1 while a Telnet client is attached.
- Less network housekeeping. It no longer runs after every statement when WiFi is not connected, and runs once a millisecond when it is: programs run about 1.5% faster. WEB SCAN finds networks when WiFi is not connected.
- TLS uses the PSRAM. The TLS handshake's working memory comes from the PSRAM on the boards that have it (WebMiteRP2350 and PicoMiteHDMIWEB with a PSRAM, and the PicoMitePC3), leaving the internal heap to the program.

Fixed

- Web server: it could close a connection it had just accepted, before the request arrived, or in the middle of sending a

page, because a time comparison wrapped. The browser got an empty reply, or the program stopped with "No response to request from connection no. n".

- Web server: it gave up waiting for a client's acknowledgement after a fixed number of loop passes, about a second on an RP2350 at 315 MHz, so WEB TRANSMIT FILE of a larger file to a Windows client usually stopped part way with "Warning: LWIP send data timeout", and the program then stopped with "Invalid parameter". The server now waits until 5 seconds pass with no progress, and after a timeout WEB TRANSMIT FILE returns with the file and the connection closed and the program carries on.
- WEB TRANSMIT PAGE: a {expression} over 31 characters overwrote memory, and a missing } read past the end of the file.
- A full connection table no longer writes past the end of the connection arrays.
- TCP server and Telnet: a connection that was reset, aborted or timed out could be closed again after the network stack had already freed it, including by a later RUN, NEW or EDIT. A Telnet client that disconnected stayed connected on the board until the board next wrote to it, and Telnet output was sent four bytes at a time from a buffer that was refilled before it had gone. These are fixed.
- A WEB TCP INTERRUPT or WEB UDP INTERRUPT handler outlived its program. A web request or UDP datagram that arrived while a later program ran made that program jump to where the old handler had been, in the middle of its own code ("Invalid character", or worse), even though it had armed no handler. Both are now cleared at RUN and NEW, as the MQTT interrupt already was.
- WEB TCP INTERRUPT 0 and WEB UDP INTERRUPT 0 stopped with "Line number" instead of switching the interrupt off. Both switch it off now.
- WEB UDP followed by anything but INTERRUPT or SEND was accepted with no error, so a typo such as WEB UDP SNED ... sent nothing. It is now "Invalid syntax", like every other unknown WEB word, so a program with such a typo now stops.
- WEB TCP CLIENT REQUEST and WEB TCP CLIENT READ into an integer array of 8192 elements or more (64 KB) received nothing ("No response from server") or lost part of the reply. Arrays of any size now work.
- WEB TCP CLIENT REQUEST, READ and STREAM could write 8 bytes past the destination array, and a timed-out WEB TCP CLIENT REQUEST wrote into the program's array after the error, which restarted the board a few seconds later. WEB SCAN and the receive commands now reject an array too small for any payload, and the CRLF ending a transmitted HTTP body is in the right order.
- An HTTPS download through WEB OPEN TLS STREAM or WEB OPEN TLS CLIENT stopped for good once the server sent its first full-size (16 KB) TLS record, because the TCP receive window, 11,680 bytes, was smaller than one record. The window is now 23,360 bytes, at no cost in memory. On the RP2040 a server's full-size records still end the connection, because its TLS input buffer is 8 KB.
- JSON$: every array index after the first was read as 0 (a[1].b[2] returned a[1].b[0]); a path may now start or end with an index ([1].x, list[2]); a key or index longer than its buffer is "Invalid JSON path" instead of an overrun; a string value over 255 bytes is "String too long"; the argument count is checked; and the parser's nesting limit is 128 rather than 1000.
- JSON$ (WebMite, PicoMiteHDMIWEB and PicoMitePC3): an error inside the call after the parse (a second argument that is not a string, say) left its working memory held, and after RUN, NEW or LOAD the next JSON$ could free memory that by then belonged to the program's variables. The arguments are now read before the parse, so no error can leave it allocated.
- WebMite, WebMiteRP2350 and PicoMiteHDMIWEB: the first WiFi join after a CPU RESTART could fail with a link failure on some routers (seen with a FritzBox), although a second attempt a moment later succeeds. The join now tries that error again, up to three tries a second apart; any other error still fails at once.
- WebMite, PicoMiteHDMIWEB, the Bluetooth host builds and the PicoMitePC3: the WiFi chip's SPI clock reached the chip's 50 MHz maximum at 200 MHz (the WebMite's default speed) and 47 MHz at 378 MHz. It is now at most 40 MHz, at boot and after CPU SPEED or RESOLUTION.
- PicoMiteHDMIWEB: MM.DEVICE$ said WebMite, and RAM FILE LOAD wiped the SUB table.
- WebMiteRP2040 (Pico W): OPTION POWER drives GP23, which on the Pico W is the radio's power-on pin, so OPTION POWER PFM switched the radio off. It now refuses with GP23 is the wireless chip's power-on pin.

Bluetooth

The Bluetooth host builds (the PicoMiteRP2350BTH, and the new PicoMiteHDMIBTH and PicoMitePC3) can now play through Bluetooth speakers and headphones, take Bluetooth Classic keyboards as well as BLE ones, and use a Bluetooth serial device as COM7, with a set of BLUETOOTH commands to find, pair and manage them. BLE keyboards pair and reconnect more reliably, and the PicoMiteRP2350BT has a serial console beside its Bluetooth one.

New and changed

Speakers and headphones

- OPTION AUDIO BLUETOOTH [idle] makes Bluetooth the board's audio output. Everything PLAY does (tones, PLAY SOUND, WAV, FLAC, MP3 and MOD files) goes to the speaker or headphones. With no speaker connected the sound is thrown away at the speed it would have played, so a program runs as it does with a wired output and nothing plugged in. The stream to a speaker starts as soon as it connects and is suspended after idle seconds of silence (0 to 250, default 60); the next sound then waits about half a second while it opens again. 0 keeps it open for as long as a speaker is connected. Encoding the sound takes 12 to 14% of the processor at 200 MHz while the stream is open. The manual covers it under Sound Output (Bluetooth Speakers and Headphones) and in the Bluetooth Versions chapter.
- The board remembers the last speaker it played through. With OPTION AUDIO BLUETOOTH set it connects to it again by itself a few seconds after a reset, and whenever it comes back in range, until BLUETOOTH DISCONNECT.
- Only 44.1 kHz is offered to a speaker. A speaker or car radio that opens the stream itself picks the sample rate from what the board offers; one that picked 48 kHz played with a gap every 3 seconds. Every A2DP device must take 44.1 kHz.

Finding, pairing and checking devices

- The BLUETOOTH commands. BLUETOOTH SCAN lists the devices in range (speakers as Audio, keyboards as Keyboard). BLUETOOTH CONNECT name$ pairs with one and connects to it by the name the scan showed (in any case) or by its address. BLUETOOTH DISCONNECT lets it go and BLUETOOTH STATUS reports the connections. BLUETOOTH FORGET deletes every stored pairing, keyboards and speakers; BLUETOOTH FORGET addr$ deletes one device's. MM.INFO$(BLUETOOTH SPEAKER) and MM.INFO$(BLUETOOTH KEYBOARD) give the connected device's address, or an empty string.
- BLUETOOTH CONNECT name$, pin$ gives the PIN to a speaker or a Classic serial device that pairs the older way (from before Bluetooth 2.1) and does not take 0000. The VDO TR723UB-BU car radio needs 1234, and so do many serial modules. Without pin$, 0000 is used. A PIN cannot be given for a keyboard or a BLE device.
- BLUETOOTH CONNECT SPEAKER | SERIAL | KEYBOARD name$. Without a word in front of the name, BLUETOOTH CONNECT chooses what a device is by its Class of Device: a keyboard, an audio device, or anything else, which is asked for a serial port and otherwise tried as a speaker, so a device that is two things at once gets only one of them. SPEAKER connects the device as a speaker without asking it for a serial port. SERIAL connects its serial port as COM7, or stops with "The device has no serial port". One of each gives both at once: an RN52 module, for example, is a speaker with a serial port, and BLUETOOTH CONNECT SPEAKER "RN52-98F5" followed by BLUETOOTH CONNECT SERIAL "RN52-98F5" gives both; each is remembered and reconnected. KEYBOARD is for a Classic keyboard whose Class of Device does not say it is one; on a device that is not a keyboard the connection is refused and BLUETOOTH CONNECT reports an error. SPEAKER and SERIAL take a PIN. A BLE device cannot be a speaker, and BLE keyboards pair by themselves, without BLUETOOTH CONNECT; both are reported. A string variable called serial$ or speaker$ is still a name.
- BLUETOOTH STATUS for a speaker shows its packet size (MTU) and the frames in each packet, the quality in use (bitpool 41 (of 53)) and how often it was lowered, packets the radio refused, and how often the stream started, paused (and how many of the pauses the speaker made), was closed by the speaker, could not be started, and was reopened. If a speaker misbehaves, a BLUETOOTH STATUS listing taken while it plays, or straight after a dropout, shows which of these it is; please send us one.

Keyboards and mice

- Bluetooth Classic keyboards. A Bluetooth keyboard no longer has to be a BLE one. An older Bluetooth Classic keyboard, and the touchpad or mouse built into one, works alongside a BLE keyboard and a Bluetooth speaker, all three connected at once. Put the keyboard in pairing mode and run BLUETOOTH SCAN, where it is listed as a Keyboard; BLUETOOTH CONNECT "name" (the name the scan showed, or the address) pairs it. If the keyboard asks for a code, the console shows Bluetooth keyboard pairing: type NNNNNN on the keyboard being paired, then Enter, as it does for a BLE keyboard. After that the keyboard reconnects by itself whenever it wakes up or is switched on, also after the board restarts. The keys go through OPTION KEYBOARD like a USB or BLE keyboard's: the layouts, the lock keys, auto-repeat and KEYDOWN() all work. A touchpad or mouse is DEVICE(MOUSE 2, ...), the same as a BLE mouse. BLUETOOTH STATUS has a Classic keyboard: line (the address, whether it is connected, and which of its reports carry the keys, the pointer and the media keys), MM.INFO$(BLUETOOTH KEYBOARD) gives its address when no BLE keyboard is connected, and BLUETOOTH FORGET forgets it.
- Tablet keyboards. Many low-cost keyboards sold for tablets have an operating-system mode, chosen with an Fn key combination or a switch. Use its Windows or PC mode: in its iOS mode such a keyboard sends Cmd+arrow for Home and End and iPad shortcuts instead of F1 to F12, and its touchpad changes too.
- BLE keyboards reconnect and pair more easily. A paired keyboard connects again after a reset with a key press, without its pairing button, and when it wakes up it uses its pairing instead of pairing afresh each time. A keyboard that asks for a passkey when it pairs could not pair before; the board now shows the six-digit code on the console, to be typed on the keyboard. This has not yet been tried with such a keyboard.

Serial devices on COM7

- A Bluetooth serial device can be the serial port COM7, much as a USB serial device is COM3 to COM6. It can be a Bluetooth Classic device with a serial port (the Serial Port Profile), such as a Bluetooth GPS receiver, or a Bluetooth LE device with a serial service: the Nordic UART Service (many modules, and the console of a PicoMiteRP2350BT), an HM-10 module's FFE0, or Microchip's Transparent UART. It has been tried with a Dual XGPS150 GPS receiver (Classic) and a PicoMiteRP2350BT's console (BLE). The Bluetooth Versions chapter of the manual and PicoMiteBT.pdf describe it in full.
- Finding and connecting it. BLUETOOTH SCAN lists a Bluetooth LE serial device as BLE serial; a Classic one is listed without a type. BLUETOOTH CONNECT "name" (or the address) connects it: Connected to serial device 00:19:01:2D:DC:BF (RFCOMM channel 1) as COM7. A Classic device is paired as it connects, answering a PIN request with 0000; a Bluetooth LE one is used without pairing unless it asks. The device is remembered, one at a time, and is connected alongside a BLE keyboard, a Classic keyboard and a speaker.
- Using it. OPEN "COM7:" AS #1 or OPEN "COM7:" AS GPS, then use it as any serial port: a buffer size, an interrupt and its level work as for COM1, a baud rate is accepted and ignored, and PRINT # ends lines with CR and LF. With AS GPS the GPS() functions decode it, every sentence of every second included.
- Staying connected. While COM7 is open the device is reconnected whenever it is not connected: when the port is opened, after it is switched off and on, and when it comes back into range. CLOSE leaves the link up. BLUETOOTH DISCONNECT SERIAL lets it go until COM7 is next opened or the next BLUETOOTH CONNECT; BLUETOOTH FORGET forgets it. BLUETOOTH STATUS has a Serial: line with the bytes received and sent, and MM.INFO$(BLUETOOTH SERIAL) gives the connected device's address, or an empty string, so a program can wait for it.
- GPS receivers. Some have a switch for use with an iPhone or iPad that hides their serial port: set it for other devices (on the Dual XGPS150, towards the USB connector). An older GPS can report a date 1024 weeks too early, a fault in its own firmware (the GPS week rollover).

The PicoMiteRP2350BT

- A serial console beside the Bluetooth one. This build has no USB, so a board whose Bluetooth link could not be used had no console at all. After a fresh install or any OPTION RESET the console now also runs on GP8 (TX) and GP9 (RX), UART1 at the console baud rate; on a Pico Computer 3 these pins go to the board's own USB-serial bridge. OPTION LIST shows it as OPTION SERIAL CONSOLE COM2,GP8,GP9,BOTH. To set it by hand use

OPTION SERIAL CONSOLE GP8, GP9, B, and keep the B: on this build it means Bluetooth and serial, and without it the Bluetooth console goes silent.

Fixed

- PicoMiteRP2350BTH: a Logitech Pebble Keys 2 K380s paired and connected but never typed, because its key report is 7 bytes, not the usual 8; the board now reads each keyboard's report layout from the keyboard itself. And while another Bluetooth LE keyboard was paired with the board, a keyboard that starts the pairing itself (the K380s again) could not pair at all. It now pairs.
- PicoMiteRP2350BTH: when another device used the board's Bluetooth LE services, the board could disconnect the keyboard. Only the keyboard's own link counts now.
- PicoMiteRP2350BTH: a Bluetooth keyboard's Caps Lock LED came on for a moment and went out again, although Caps Lock stayed on, because only USB keyboards were sent their lock LEDs. Bluetooth keyboards now are, for Caps, Num and Scroll Lock.
- PicoMiteRP2350BTH: a Bluetooth keyboard pairing or waking up while sound played could pause the sound and make an SD card read fail with SDcard Removed, because every save of its pairing held everything up for about 160 ms. The pairings are now saved once the keyboard has finished.
- PicoMiteRP2350BT and PicoMiteRP2350BTH: at boot Bluetooth set up its pairing store in the last 12 KB of flash, which is the end of the A: drive, and erased a 4 KB sector there, which could damage a file stored there. The pairings now live with the options, and Bluetooth writes nothing to the A: drive.
- GUI CURSOR LINK MOUSE stopped with "No mouse connected" on the Bluetooth host builds when the mouse was a Bluetooth one (a BLE mouse, or a BLE or Classic keyboard's trackpad), though such a mouse moves DEVICE(MOUSE 2) as a USB mouse does. It is accepted now.
- PicoMiteRP2350BT: the console's Bluetooth LE scan response did not carry the Nordic UART Service's identifier, so anything that looks for the console by its service (a PC scanning with bleak, or the Bluetooth host builds' BLUETOOTH SCAN) did not see it. Phone terminal apps connect by name and were not affected.

Tools and documentation

mmb2csub compiles a BASIC SUB or FUNCTION into a CSUB, for the PicoMite or the Colour Maximite 2, without writing C; the CSUB interface and its header, PicoCFunctions.h, have grown; and there are new manuals and corrections to the old ones.

New and changed

- mmb2csub. A tool, in its own zip, that takes one SUB or FUNCTION out of your program, compiles it to machine code and puts it back as a CSUB; your call sites do not change. Loops, arithmetic and array work run 10 to 35 times faster: the Julia set demo renders in 5.9 s instead of 119 s on an RP2040. The generated code calls the firmware's own routines, so SIN, MID$ or RGB inside the CSUB mean exactly what they mean in BASIC, and one converted program, the same bytes, runs on the RP2040 and the RP2350 and on every variant. --list reports what every routine in a program would gain before anything is written, and anything unsupported is refused by name. --target cmm2 builds CSUBs for the Colour Maximite 2.
- More in the CSUB CallTable. New vectors, all appended so existing CSUBs keep working: the 64-bit integer helpers LMul, LDiv, LMod, LShl, LAsr and LLsr; memcpy, memset and memmove; the MMBasic string routines MtoC, CtoM, Mstrcpy, Mstrcat, Mstrcmp, StrRight, StrCase, StrMid, StrChar and StrFill; INSTR, STR$, TIMER, RND, log, tan, VAL and TIMER = n; the current COLOUR foreground and background; and PinVal and PinPut, which read and drive a pin as PIN(n) and PIN(n) = v do (20,000 write-then-read passes in 83 ms rather than 654 ms). They sit at 0x140 to 0x1BC, and the CSUB manual (armcfgen) lists every vector with its signature.
- Sixteen CSUB arguments on the RP2350; ten on the RP2040, as before.
- One CSUB for both chips. The CallTable is found at run time, so one compiled CSUB runs on the RP2040 and the RP2350 and on every variant; a CSUB built with the new header needs this release or later. The exception is a CSUB that reads the variable table (its dims[] are five ints on the RP2350 and six shorts on the RP2040) or the Option structure (laid out per variant): that must still be built for the chip, or variant, it runs on. The CSUB manual

says which CSUBs are tied to one chip; all mmb2csub output runs on every board.
- PicoCFunctions.h v2.3.0. Anyone compiling a CSUB in C or assembler should take the PicoCFunctions.h from this release (it is in the repository and inside the mmb2csub zip) and rebuild. It declares the new vectors, and its Option structure matches the firmware's 896-byte one, including the new LIBRARY_HASH field.
- New manuals. The user manual covers everything in this release. A PLAY BBC manual, the CSUB manual (armcfgen) and the mmb2csub manual are new.
- MMBasic Performance Guide. A new companion manual, MMBasic_Performance_Guide.pdf, answers the common "how do I make it faster" questions with measurements on an RP2040 PicoMiteVGA and an RP2350 PicoMiteHDMIWEB, interpreted and compiled: statements per line, FOR against DO, constants against variables, CALL by name, long names, and when the MATH, ARRAY, SORT and MEMORY commands pay. It is in the supplementary manuals zip.
- PicoMiteBT.pdf, now titled "PicoMite Bluetooth Versions", covers the two Bluetooth host builds as well as the PicoMiteRP2350BT: keyboards, mice and game controllers, speakers and headphones, pairings and BLUETOOTH STATUS.

Fixed

- PicoCFunctions.h declared the wrong variable-table layout for the RP2040: a CSUB built from the old header against an RP2040 build read every variable's value from the wrong offset. MAXDIM is now 5 on the RP2350 and 6 on the RP2040.
- The user manual had MM.INFO(FONT ADDRESS n) and MM.INFO(FONT POINTER n) the wrong way round: FONT ADDRESS is the address of the font, and FONT POINTER the address of the font table's entry for it, as they have always been. It now also says what a font's first four bytes are, and that DEFINEFONT takes fonts 2 to 5 and 8 to 16.
- The documentation did not say that LOCATION's time is UTC, not local time. A local time made every ASTRO result wrong by the time zone's offset, so a moonrise found from ASTRO's altitude came out an hour early in central Europe in winter. GPS_Astro_Reference.pdf, the manual and HELP now say so, with an example of the conversion; STAR uses the GPS's UTC time, whatever offset OPEN ... AS GPS was given.
- The manual said OPTION KEYBOARD ignored everything after the layout for USB and Bluetooth keyboards. Its caps lock, num lock and repeat settings work for those keyboards too, as they always have.
- The manual's command list was missing DO UNTIL condition ... LOOP and LOOP WHILE; both have always worked.

Appendix: what V6.03.01 changed from V6.03.00

For anyone still on V6.03.00, V6.03.01 (4 August 2026) added these, all of which V7.00.00 keeps.

New

- WEB PING (WebMite and PicoMiteHDMIWEB): WEB PING address$ [, count] [, timeout] [, avgvar] sends ICMP echo requests to a host name or IP address, prints a line per reply and a summary, and can return the average round-trip time in ms (-1 if there was no reply) in a floating point variable.
- OV5640 camera (RP2350): CAMERA OPEN OV5640 gives a 320x240 RGB565 preview, motion detection (CAMERA CHANGE) and JPEG capture (CAMERA CAPTURE JPEG) at VGA, XGA, SXGA, UXGA, QXGA and 5MP, with HIGH, MEDIUM and LOW quality.
- A camera without a display: CAMERA OPEN no longer needs a panel. CAMERA CAPTURE JPEG and CAMERA CHANGE work without one, and the simple CAMERA CAPTURE takes any display with at least RGB565 colour, SSD1963 panels included.
- TURTLE takes the COLOR spelling: PEN COLOUR, PEN COLOR, PENCOLOUR, PENCOLOR and PC are all the same, and likewise FILL COLOUR and CURSOR COLOUR.
- USB mice decoded from their own HID report descriptor when OPTION MOUSE SENSITIVITY is set, so mice with extra buttons, padding, report IDs or wider movement fields work.
- The Pico Computer 3 is detected and configured automatically at boot (RP2350).

Fixed

- CIRCLE with an aspect greater than 1 and a line width of 1 left gaps in the outline.
- A mouse click or screen tap in the full screen editor landed on the wrong character on a line that had scrolled sideways.
- A camera that was misconfigured or disconnected froze the board; it now gives an error naming the signal that stalled. Camera capture no longer disturbs the PSRAM and flash controller, which caused rare hard faults on PSRAM builds.
- Bluetooth console and keyboard fixes; the pin-status program built into the WEB builds and the OLIMEXUSB platform name corrected; keyboard map updates.