Arrays of One

MMBasic now accepts a dimension holding exactly one element. `DIM a(0)` under `OPTION BASE 0`, and `DIM a(1)` under `OPTION BASE 1`, used to raise *Dimensions*; they now declare a one-element array. Nothing that worked before changes — the release only makes previously illegal declarations legal.

The gap this closes

Until now MMBasic could describe a variable with no dimensions at all — a scalar — and a dimension of two or more elements, but not a dimension of exactly one. Every cardinality was available except the smallest.

The hole was never a language decision. `dimtbl[]` stores each dimension's declared upper bound, and the value 0 was already doing three other jobs: it marks "not an array", it terminates the dimension list, and it drives the `ERASE` heap-ownership test. Under `OPTION BASE 0` a one-element dimension has an upper bound of 0, so it had nowhere to live. The change reserves `DIM_ONE (-2)` for that single case and decodes it at every read.

What the programmer gains

A calculated size is allowed to come out at one

Array sizes come from data: a record count, a channel count, a parsed field count, the size of a filtered subset. Any of those can legitimately be one. A program that was correct for every other value failed on the smallest — and that is the value test data almost never has, so the failure tended to show up in the field rather than on the bench.

WAS

```
' n comes from the data
n = CountRecords()
DIM rec(n - 1)    ' n=1 -> Dimensions

' so you wrote this instead
IF n = 1 THEN
   ' a whole second code path
ELSE
  DIM rec(n - 1)
END IF
```

NOW

```
' n comes from the data
n = CountRecords()
DIM rec(n - 1)    ' any n >= 1

' and that is the whole program
```

Degenerate matrices become representable

A 1×N row, an N×1 column and a 1×1 result are ordinary outcomes of matrix work, not error cases. Their absence bit hardest in the `MATH` commands, where the dimension-existence test asked whether a dimension had at least *two* elements — the wrong question, and wrong under either option base.

WAS

```
DIM a(3,0)       ' Dimensions

' and with the shape forced
' into existence by hand:
MATH SLICE a(), , 0, col()
                 ' Argument count
```

NOW

```
DIM a(3,0)       ' 4 x 1

MATH SLICE a(), , 0, col()
                 ' the whole column
```

One interface instead of two

A `SUB` or `FUNCTION` that takes `arr()` now serves a collection of one without a wrapper, a scalar overload, or the old trick of declaring two elements and passing the real count alongside. Library code stops carrying a special case that existed only to satisfy the declaration rule.

WAS

```
DIM pts(1)       ' 2, but only
count = 1        ' 1 is real
Plot pts(), count
```

NOW

```
DIM pts(0)       ' exactly 1
' BOUND() is now the truth
Plot pts()
```

Growth can start where the data starts

A buffer that grows with `REDIM PRESERVE` can now begin at its true size of one, rather than being declared at two and shadowed by a separate fill counter that the rest of the program has to remember to consult.

WAS

```
DIM buf(1)       ' minimum legal
used = 1         ' track it yourself
' ... later
REDIM PRESERVE buf(n)
used = n + 1
```

NOW

```
DIM buf(0)       ' 1 element
' ... later
REDIM PRESERVE buf(n)
```

Structures and generated code stop needing a shape check

A `TYPE` can carry a one-element member array, and a single structure can be declared through the array syntax. Code that *writes* BASIC — importers, exporters, table generators — can emit whatever the data says without a branch for "if the count came out at one, emit something structurally different".

WAS

```
TYPE pt
  vals(0) As integer   ' Dimensions
END TYPE

DIM arr(0) As pt    ' Dimensions
```

NOW

```
TYPE pt
  vals(0) As integer   ' ok
END TYPE

DIM arr(0) As pt    ' ok
```

Both option bases behave the same way

The restriction was not a base-0 quirk: `DIM b(1)` under `OPTION BASE 1` was refused too, which reliably surprised anyone arriving from a base-1 dialect. Both bases now accept a one-element dimension, and under base 1 the new encoding is not even involved — an upper bound of 1 is stored as itself.

WAS

```
OPTION BASE 1
DIM b(1)        ' Dimensions
DIM b(2)        ' the minimum
```

NOW

```
OPTION BASE 1
DIM b(1)        ' 1 element
DIM b(0)        ' still refused
```

The hardening dividend

Making the change meant routing roughly 230 dimension reads across sixteen files through a single accessor layer. That audit turned up defects with nothing to do with one-element arrays, which are fixed and stand on their own merits:

- `MATH INSERT` and `MATH SLICE` tested whether a dimension existed with `dims[i] - OptionBase > 0`. That asks whether the dimension has at least two elements — not the same question, and wrong under either base.
- `WEB SCAN` and the three `WEB TCP CLIENT` receive paths wrote past the end of an array too small to hold a payload, because element 0 is the length word. They now report *Array too small*.
- A row leak in `MATH M_INVERSE`, an unguarded degrees-of-freedom index into the chi-square critical-value table, and an over-read in the PIO buffer sizing were found and fixed along the way.
- `LIST VARIABLES` silently dropped a trailing single-element dimension from its output; it now prints the declared shape.

Cost and compatibility

DIMENSION	POSITION
Existing programs	Unaffected. Only declarations that previously raised an error become legal; no working construct changes meaning.

DIMENSION	POSITION
Flash	+1–2 KB across the nine build variants. The tightest, HDMIUSB, keeps +7.6 KB of margin.
RAM	Roughly 256 bytes, with +3.4 KB of margin remaining on the tightest variant.
Reversibility	Setting `DIM_DECODE_ENABLED` to 0 collapses every accessor back to the expression it replaced and stores no escape, restoring the old behaviour exactly. All nine variants built byte-identical against the pre-change firmware at that setting, which makes it a clean bisect point.
CSUB ABI	Untouched. The one caveat: a CSUB that reads the dimension table directly will see `-2` for a base-0 one-element dimension.
`BOUND()`	The single deliberate behaviour change. 0 is now a real upper bound, so it can no longer also mean "no such dimension": asking for a dimension the variable does not have is an error rather than a silent 0.

Validated on hardware

Exercised on a WebMite RP2350B running the HDMIWEB build, over both option bases and all three element types:

`1-D, (3,0), (0,3), (0,0)` `float / integer / STRING LENGTH` `index + bounds errors`
`BOUND()` `MATH SLICE` `MATH INSERT` `SUM / MAX / MIN / MEAN` `SORT` `MATH SCALE`
`DIM initialisers` `SUB parameters as arr()` `LOCAL / STATIC` `VAR SAVE / RESTORE`
`REDIM` `REDIM PRESERVE` `struct member arrays` `WEB SCAN guard`
`200 DIM/ERASE cycles, no leak` `regression pass on ordinary arrays`

Still refused, exactly as before: `DIM a(-1)` under base 0, and `DIM b(0)` under base 1.

Bottom line

> The two-element minimum was an artefact of how the upper bound was stored, not a statement about what an array is. Removing it deletes a class of bug that only ever appeared on the smallest input, costs one to two kilobytes of flash, and leaves every existing program running exactly as it did.

Implementation: phase A accessor layer (8b4abd4) · phase B1 local-copy consumers (23fef04) · phase B2 feature enable (d2a22dc) · hardware validation (4563da6).
Site inventory and test matrix: docs/dims_phase_a_audit.md