# TM1637_Scroll — Reference (EN)

A Flowcode component for scrolling text on a TM1637 display (4-digit, 7-segment). Non-blocking, 7-segment font, number conversion, custom characters, adjustable speed. Built on top of the standard TM1637 component (uses its `ShowSegments` internally).

---

## 1. Project setup

| Step | What to do |
|---|---|
| Pins | Select the component in the panel → set `DATA_PIN` and `CLK_PIN` in Properties |
| Interrupt | Add a Timer interrupt in your project at **1000 Hz** (1 ms) |
| Link | The interrupt calls a project macro `OnTick`, which calls `Tick()` |

> WARNING: The interrupt lives in the PROJECT, not in the component. The component receives its "pulse" through the `Tick()` macro. This keeps the component timer-independent.

Project macro `OnTick`:

TM1637_Scroll1.Tick()

> WARNING: Don't call `Tick()` directly from the ISR — it must go through a project macro, otherwise you get an "undefined symbol" link error.

## 2. Main loop

[loop While 1]

  // rebuild the text ONLY when the value changes

  [Decision] val != oldVal ?

    Yes -> BufClear() ... BufFinish() + oldVal = val

  TM1637_Scroll1.ScrollStep()

## 3. Commands — building the buffer

Call these between `BufClear()` and `BufFinish()`. Each call is a Component Macro icon.

| Command | Parameter | What it does |
|---|---|---|
| `BufClear()` | — | Resets the buffer. Always first. |
| `BufAppendSpace(count)` | number | `count` blank positions |
| `BufAppendChar(c)` | char / ASCII code | One character from the font |
| `BufAppendNum(n)` | number | An integer, digit by digit |
| `BufAppendSeg(raw)` | segment byte | A custom character (raw byte) |
| `BufFinish()` | — | Trailing padding + finalize. Always last. |
| `ScrollStep()` | — | Advances the scroll (in the loop) |
| `SetSpeed(val)` | ms/step | Sets speed (higher = slower) |
| `Tick()` | — | Increments the internal counter (from the ISR) |

### Example: "TENP 85°C"

```
BufClear()

BufAppendSpace(4)

BufAppendChar('T')

BufAppendChar('E')

BufAppendChar('N')      // 'M' is impossible on 7-seg

BufAppendChar('P')

BufAppendSpace(1)

BufAppendNum(temp)

BufAppendSeg(99)        // degree

BufAppendChar('C')

BufFinish()
```

## 4. Gotchas (learned the hard way)

- WARNING: `BufAppendChar` -> single quotes `'T'` or ASCII code `84`. NOT double quotes `"T"` (string -> error).
- WARNING: `BufAppendSpace` -> NUMBER of spaces, no quotes: `4` (not `'4'` = 52 spaces!).
- WARNING: Too low a ScrollSpeed makes text unreadable — on 4 digits the scroll is character-by-character. 250–400 ms/step is comfortable.

## 5. Custom characters (BufAppendSeg)

Value = sum of segments: a=1, b=2, c=4, d=8, e=16, f=32, g=64, dp=128.

| Symbol | Value |
|---|---|
| degree | 99 |
| minus | 64 |
| low bar | 8 |
| high bar | 1 |
| equals | 72 |

## 6. Character set

- Digits: 0-9
- Uppercase: A b C d E F G H J L n o P q r S t U y
- Lowercase: a b c d e f g h i j l n o p q r s t u y
- Impossible (blank): K M V W X Z
- Expected collisions: g~~9, e~~E, S~~5, o~~0

---

## How it works (short version)

The text is converted into an array of raw segment bytes (`segBuf[]`) with blank padding at both ends. `ScrollStep()` writes a 4-byte sliding window of that array to the display via `ShowSegments`, advancing one position every `ScrollSpeed` ms. A timer interrupt (through `Tick()`) increments a millisecond counter, so the timing is non-blocking — the rest of your program keeps running between steps. The `Buf*` macros only touch the buffer; the only hardware contact is inside `ScrollStep`, so the engine is portable to other displays by swapping just the output call.

---

## Notes for builders

- The component contains the standard TM1637 as an internal sub-component — you only add `TM1637_Scroll` to your project, not a separate TM1637.
- Keep the interrupt at exactly 1000 Hz so `ScrollSpeed` is in real milliseconds.