

Technical Brief — Implementing Folio-level Mirror and Flip in QET

Feature: non-destructive Mirror and Flip of placed element instances at folio (diagram) level, composable with Rotate. **Branch:** `folio-mirror-flip` (fork: `github.com/hairykiwi/qelectrotech-source-mirror`), built against `qt6_cmake_joshua`. **Status:** implemented, tested, ready for review. **Author:** hairy_kiwi. Diagnostic and verification work carried out with Claude AI + Claude Code.

This brief accompanies the demo project (`.qet` + PDF). The demo shows *what* the feature does; this document explains *how* it works, the one substantial bug found and fixed along the way, and how the fix was verified. It is written for a reviewer who may want to merge the branch.

How the feature is stored

Mirror and Flip are represented as two boolean flags on the placed element, serialised as attributes on the element's `<element>` entry in the `.qet` file. They are **not** baked into geometry: the saved coordinates remain the element's design coordinates, and the mirror/flip is re-applied as a live transform when the diagram loads. This has two consequences worth noting for review:

First, it is backward compatible. Older QET versions that do not recognise the flags simply ignore them, so a file saved by this branch still opens in earlier releases (the element just renders un-mirrored). Files created by earlier versions load here unchanged.

Second, the transform is non-destructive. Because the mirror lives as a flag rather than as altered coordinates, it can in principle be cleared to return the element to its on-disk orientation — the basis for the proposed "transform reset" follow-up.

The orientation model

A placed element carries a rotation and the two new flags. The effective on-screen transform is the composition of the element's rotation with the mirror and/or flip reflection, applied about the element's reference point. Mirror is a reflection of determinant -1 ; flip is likewise determinant -1 ; the two composed give determinant $+1$ (a pure 180° rotation of the element frame). Rotation alone is determinant $+1$.

Dynamic element text attached to the element must remain readable after the transform — text should not end up reflected (mirror-written) when the element is mirrored. The feature therefore applies a compensating transform to each text item so that the *element geometry* reflects while the *text* stays upright and correctly placed. The determinant of the text's resulting transform is the diagnostic used throughout verification: a correctly handled text item keeps the same orientation parity as its parent frame, with no stray extra reflection.

The bug: grouped rotated text corrupted on save / reload

The substantial issue was not in the live transform — which rendered correctly on screen — but in serialisation. An element can carry a *group* of dynamic text lines (an `ElementTextItemGroup`), and each line in the group may itself be rotated. When such an element was mirrored and then saved and reloaded, the grouped text was displaced by one group-width and reflected, even though it had looked correct before saving.

Mechanism

Text in a group is stored child-relative: each line records its offset from the group's origin, and the group records its own position and rotation. The on-screen position is the composition of the two.

The save routine did not write each child's stored offset directly. Instead, for each child it called `ElementTextItemGroup::removeFromGroup()` — a routine whose actual purpose is to *genuinely ungroup* text, detaching a line from its group while preserving its on-screen position by recomputing its coordinates into the parent frame. That position-preserving recomputation is correct for a real ungroup.

The save path, however, borrowed this routine purely to serialise grouped text, with no genuine ungroup intended. On a mirrored element the group's transform carries the mirror reflection, and the position-preserving recomputation therefore folded that reflection into the child's stored x/y coordinates. The mirror became half-applied: baked into the saved position, while the element's mirror flag was still present to be applied afresh on load.

On reload, the loader read the (now reflection-baked) child coordinates **and** re-applied the element's mirror flag, applying the mirror twice. The doubled reflection displaced the text by one group-width — the observed corruption.

Why the determinant was not the discriminator

An early hypothesis was that the corruption could be detected by the sign of the compensation transform's determinant. This proved false: the determinant was -1 for both mirror and flip, and -1 also appeared in passing (uncorrupted) ungrouped rotated cases. The determinant sign alone did not separate good from bad states. The true signal was a *parity split*: at the moment of serialisation, the group's scene-transform determinant and its children's determinants disagreed ($+1$ against -1) specifically in the corrupting case, indicating a reflection present in one frame but not consistently in the other. That split — not the absolute determinant — located the fault at the save path rather than the load path.

The fix

The corrupting detour was removed. Each grouped child's geometry is now serialised directly, in a single frame: the child's design-relative position is computed from the group's own position and rotation and the child's stored offset, and the child's rotation is written as the plain group rotation. No `removeFromGroup()` call is made during serialisation, so no reflection is folded into the saved coordinates, and the live scene is left untouched. The mirror remains solely a flag, recomputed on load — one source of truth instead of two.

Genuine ungroup (a user actually ungrouping text) still uses `removeFromGroup()` unchanged; that path was deliberately not modified, since its position-preserving behaviour is correct for its intended purpose.

A non-fatal desync tripwire (a `qWarning` plus an invariant comment) was added at the serialisation site to surface any future divergence between the live and serialised group geometry, without aborting on transient states.

Verification

The fix was verified against a matrix of transform cases, each exercised with the same protocol: perform the transform, save (capture A), close, reopen, eyeball the result, save again with no further action (capture B), then compare A and B on disk and grep the instrumentation. Three independent criteria had to hold for a pass:

1. orientation parity (group and child scene-transform determinants agree, both +1);
2. idempotency (the second save is byte-identical to the first after filtering the volatile save-metadata fields — `savedfilename`, `savedfilepath`, and `savetime`, two of which change on every save); and
3. a clean visual on reload.

Case	Transform	Result
Acceptance	mirror, 2-line group rot 45° + 3-line group rot 315°	pass
C	flip, grouped	pass
E90	element rotation 90° + mirror	pass
E180	element rotation 180° + mirror	pass (one benign pre-settle transient)
F	mirror + flip together	pass
D-mirror	covered by the 3-line acceptance group	pass
D-flip	covered by construction (same path as C)	—

Every case that exercises the changed save path restored orientation parity (group determinant equals child determinant equals +1, the original corruption signature gone), round-tripped without the one-group-width drift, and produced a null on-disk diff between the two saves after filtering the volatile save-time fields. The E180 case showed a single momentary transient: immediately after the mirror was applied, the group and child linear parts differed in sign pattern while both retained determinant +1; this resolved before the save, round-tripped identically, and was confirmed stable on a second reopen — a cosmetic settle artifact, not a data error.

On-disk diffs in multi-element folios showed element-reordering and UUID regeneration as expected non-substantive churn; per-element geometry was confirmed identical across the round-trip in every case.

Known issues and scope

Two display matters remain under investigation. Both concern only the *live, unsaved* rendering of grouped rotated text; in every observed case the saved file is correct, and once the file is saved and reopened the element displays and prints / exports correctly in its last-saved orientation. Neither affects persisted data.

Grouped rotated text under plain rotation. When an element containing grouped rotated text is rotated, the grouped lines render correctly only when the element's rotation differs from its last-saved orientation by an even multiple of 90°; at odd multiples (90°, 270°) the lines momentarily overlap. Saving re-establishes the current orientation as the reference, so the correctly-rendered and overlapping angles swap after a save / reload. This appears to be independent of the mirror/flip feature (it occurs under rotation alone) and is being investigated separately as a possibly pre-existing rendering issue.

Grouped rotated text under rotation combined with mirror/flip. At 0° element rotation, mirror and flip behave as expected. At 90° / 270° element rotation, the mirror/flip transform applied to grouped rotated text is not yet correct (the text may not move to the expected side, and overlapping persists); at 180° it is close but not exact. As above, the saved-and-reopened result is correct. Characterising and fixing this combination is ongoing; the likely direction is the same correction already applied to *ungrouped* rotated text under mirror/flip, extended to the grouped path. Until then, mirror/flip of grouped rotated text on a rotated element should be treated as a known limitation.

Both are low-impact in practice: grouped rotated text is itself uncommon, and the rotated-plus-mirror/flip combination more so. The feature is fully correct for the common cases — ungrouped text, rotated ungrouped text, and grouped text at 0° element rotation — across save/reload, as verified in the matrix above.

A second, deferred item shares the corrupted bug's family: a *genuine* ungroup of a mirrored element (via Properties → Delete-group) displaces the freed text by roughly one text-line height, and this displacement persists to disk. It travels the same `removeFromGroup-through-reflection` path that this fix deliberately left untouched on the serialisation side, and likely admits the same shape of fix (re-express the child geometry in one mirror-free frame rather than relying on the position-preserving detach). Its severity is low: the path is reachable only via Properties → Delete-group, as no right-click ungroup gesture exists. It is tracked separately for a future fix.

Static (non-dynamic) text within a mirrored or flipped element reflects as a graphic — i.e. it reverses — which is correct behaviour for a graphic primitive but is usually not desired for a label. Converting such text to dynamic text resolves it. The demo's non-dynamic-text section illustrates this.

Build note

The branch carries a small local change to `sources/main.cpp` that temporarily sets the application name to "QElectroTech-Qt6". This avoids a `SingleApplication` conflict with a parallel Qt5 build — which would otherwise treat the two as the same running instance (they share the application name as the single-instance key) and refuse to launch the second. It is a local development convenience, unrelated to the mirror/flip feature, and should be reverted before merge.