Persistent Modular Phase-State Updates for GFSK Waveform Generation

A Reproducible WSJT-X Case Study

Rudolf Stepan

Independent researcher, Vienna, Austria

OE3SRO | ORCID: 0009-0004-2842-2579

15 October 2026 | Journal research draft, version 0.3

Research draft; not peer reviewed.

Abstract

A digital oscillator need not keep track of completed turns. It can store phase within one period and correct that state after each frequency step. This paper examines how that familiar principle affects the GFSK waveform generators of a WSJT-X fork. The centered formulation is motivated by Stepan's REIST-Division manuscript. We establish the conditions under which one correction suffices in binary64 arithmetic, then compare centered and positive states with general remainder, masked correction, Q48 fixed point, unbounded accumulation and two block-prefix implementations. On an Intel Core i9-14900K, the general-remainder implementation takes about 2.6 times as long as centered correction on the selected performance core and 4.4–5.2 times as long on the selected efficiency core. Positive and centered states cost nearly the same. The tested fixed-point and prefix variants are slower. Numerical checks cover encoded frames, long sequences, interval boundaries and digital spectra. The practical limit is modest: the already optimized waveform generator accounts for 2.13–3.73% of the measured file-decoder time. Even removing that work entirely would increase total processing speed by only 2.17–3.87%, if all other work stayed constant. The study therefore contributes a reproducible comparison of phase representations and their application cost. It establishes neither an exclusive advantage of centered arithmetic nor improved radio sensitivity. Source hashes, raw measurements and analysis scripts accompany both language editions.

Keywords: modular phase accumulation; GFSK; DDS; reproducible benchmarking; floating-point arithmetic; WSJT-X.

1 Introduction

To generate an oscillating signal, software advances a phase angle and converts it to a sample. In Gaussian frequency-shift keying (GFSK), the frequency changes smoothly between symbols, so the phase increment also changes from sample to sample. A full turn leaves the ideal oscillator output unchanged. The program can therefore discard completed turns and retain only the phase within one period.

The useful question is how much work this requires. A general remainder function can reduce a wide range of arguments. If both the stored phase and the next increment are already bounded, a single addition or subtraction may be enough. Stepan's REIST-Division manuscript [1] describes this persistent centered-state approach for integer remainders and discusses phase windows in Section 6. Here we apply that formulation to floating-point phase. The derivation below is self-contained; earlier CPU and FPGA speedups from the REIST manuscript are not used as measurements of this application.

An earlier German report [2] compared centered correction with general remainder while holding the trigonometric method constant. The present study extends that comparison in three ways. It examines the floating-point assumptions and remaining errors, adds alternative phase representations under matched waveform conditions, and measures how much time waveform generation occupies in a complete file decoder. The processor's performance and efficiency cores are measured separately.

The main finding has two parts. Bounded updates substantially reduce waveform cost relative to general remainder on the measured machine, but centering offers little speed advantage over a positive interval. Meanwhile, waveform generation is already a small part of decoder time. These observations define the scope of the contribution: a checked implementation and a reproducible evaluation of an established state principle, including its practical limit.

2 Related work and attribution

Direct digital synthesis (DDS) commonly uses a phase accumulator followed by a phase-to-amplitude mapping. Frequency resolution and phase truncation are standard design concerns [3]. A floating-point state in $[0, T)$ is thus a natural comparison for a centered state. General remainder serves a different purpose: it accepts a broader argument range and specifies how half-period ties are resolved.

Two other bodies of work are relevant. Floating-point analysis explains why an identity over real numbers need not survive rounding and when subtraction is exact [4]. Prefix sums provide a way to compute several phase offsets together [5]; they shorten some dependencies but change the order of addition and require extra arithmetic and storage.

The designers of FT4 and FT8 describe their GFSK modulation, synchronization and error correction in [6]. We study the waveform generator used by those modes and the fork's FT2 integration. Separate experiments on LDPC decoding and synchronization are documented in the accompanying decoder dossier.

DDS research also makes clear that phase accumulation is only one part of an oscillator. Cordesses [9] discusses tuning resolution and phase and amplitude quantization. Vankka's dissertation [10] covers memory compression, Taylor approximation, CORDIC and sources of spurious output; his publications on amplitude mapping and spur reduction address these issues separately [11, 12]. Our comparison keeps the 65 536-entry interpolated amplitude table fixed to isolate the effect of changing the phase representation.

Other oscillator designs deserve a separate comparison. Palomäki and Nurmi [13] use second-order Taylor interpolation in a quadrature FPGA DDS with compressed memory. Their hardware clock rates and spur results cannot be read as CPU waveform timings or IQ error energies. Vicanek's technical note [14] describes a recursive quadrature oscillator and discusses frequency modulation. Such a recurrence updates amplitude states directly; changing GFSK frequency also requires coefficient updates. Neither approach is timed here. A useful follow-up would implement them under the same modulated inputs and accuracy requirements.

The REIST attribution refers to the supplied manuscript [1]; its SHA256 is recorded in the audit. The printed DOI and Zenodo metadata could not be independently retrieved. Reference [3] in that manuscript contains a DOI belonging to an unrelated publisher article and is not reused here. The connection to REIST rests on its state definition and additive update.

Table 1: Prior methods and the actual relationship to this study.

Method/source	Principle	Difference and empirical role
Persistent centered state, Stepan [1]	Integer state plus bounded signed correction	Motivates the application. Floating invariants and GFSK tradeoffs are evaluated here; earlier benchmark numbers are not reused.
DDS/NCO [3]	Modular phase register, amplitude mapping	Positive floating state and Q48 integer state are conventional competing implementations.
General remainder	Reduce a broadly valid input each update	Same double increments and LUT; measured reference with broader functionality.
Prefix scan [5]	Reassociate partial sums	Serial and tree block-64 architectures; additional reductions and rounding are included.
Compressed/Taylor DDS [10, 13]	Different amplitude mapping	Reviewed, not timed; the mapping is fixed in this experiment.
Recursive quadrature [14]	Update amplitude states	Reviewed, not timed; modulation and amplitude error need separate controls.

3 Mathematical foundations

3.1 The centered phase state

We first consider exact arithmetic over the real numbers. Let $T > 0$ denote one period. Two phases that differ by an integer multiple of T describe the same position within that period. To store a phase, we therefore choose one representative from each such class.

Definition 3.1 (Centered representative)**.** For a real number x, define

$$C_T(x) = x - T\left\lfloor x/T + \tfrac{1}{2}\right\rfloor \in [-T/2, T/2). \tag{1}$$

We call $C_T(x)$ the canonical centered representative of x.

Here $\lfloor\cdot\rfloor$ denotes rounding down to an integer. The formula subtracts a whole number of periods. The left endpoint of $[-T/2, T/2)$ belongs to the interval, while the right endpoint does not. This makes the representative unique, including at a half-period.

The definition applies to any real input. In an oscillator, however, the previous phase is already known. If the next increment is bounded, we can use that information to simplify the calculation.

Proposition 3.2 (One correction suffices)**.** *Let $r \in [-T/2, T/2)$ be the current state and $u \in [-T/2, T/2]$ the next increment. Set $s = r + u$. Then the update*

$$r^+ = \begin{cases} s - T & s \geq T/2, \\ s + T & s < -T/2, \\ s & \textit{otherwise} \end{cases} \tag{2}$$

satisfies $r^+ = C_T(r + u)$. In particular, the new state is again in $[-T/2, T/2)$.

Proof. The assumptions imply $-T \leq s < T$. We distinguish three cases.

1. If $s \geq T/2$, subtracting one period gives $-T/2 \leq s - T < 0$.
2. If $s < -T/2$, adding one period gives $0 \leq s + T < T/2$.
3. Otherwise, $-T/2 \leq s < T/2$, and no correction is needed.

In every case the result lies in the chosen interval and differs from s by an integer multiple of T. By uniqueness of the centered representative, this proves the assertion. □

The proposition also explains why the procedure can be repeated. If the initial state lies in the interval and every increment obeys the bound, each update returns a state to which the proposition applies again. Induction therefore gives interval membership after every update. We call this preserved property a *state invariant*. It is the additive application of the REIST formulation [1, Theorem 3.6 and Corollary 3.7].

Example 3.3. Take $T = 360$, $r = 170$ and $u = 30$. Then $s = 200$, so one period is subtracted and $r^+ = -160$. Both angles describe the same position. Degrees make the example easy to picture; the implementation uses radians.

Remark 3.4 (Positive state)**.** The same argument works for $p \in [0, T)$. With the same signed increment bound, $p + u$ lies in $[-T/2, 3T/2)$. Add one period if the sum is negative, subtract one if it is at least T, and otherwise leave it unchanged. Thus bounded state and bounded increments make general division unnecessary. Centering is not essential to that conclusion.

3.2 The floating-point update

We now turn to the arithmetic used by the program. Let $P = \mathrm{double}(\pi)$ denote the stored approximation to pi, and let $\widehat{T} = 2P$ be the represented period. Multiplication by two is exact here. Write fl for rounding to the nearest binary64 value.

The real-valued proposition is not by itself a proof about this calculation. We must also check whether rounding can move the result outside the chosen interval.

Proposition 3.5 (Preservation of the binary64 interval)**.** *Assume binary64 arithmetic with rounding to nearest and gradual underflow, finite stored values $r \in [-P, P)$ and $u \in [-P, P]$, and the stated order of operations. Set $s = \mathrm{fl}(r + u)$ and apply Eq. (2) with period $\widehat{T}$. Then the corrected state belongs to $[-P, P)$. Any corrective addition or subtraction is exact.*

Proof. Rounding preserves order, so $s \in [-2P, 2P]$. Unlike the real-valued case, the upper endpoint may be attained. For $s \geq P$, the positive values s and $2P$ differ by a factor of at most two. Sterbenz's lemma therefore makes $s - 2P$ exact, with result in $[-P, 0]$. For $s < -P$, the expression $s + 2P$ is the subtraction of positive magnitudes with the same factor-of-two property. It is exact and returns a value in $[0, P)$. In the remaining case, s already belongs to $[-P, P)$ and is left unchanged. This proves the assertion, including the endpoint $s = 2P$. □

We next consider error. For a normal sum, the initial addition is the only operation in this update that rounds. Its absolute error is at most $\epsilon_u \widehat{T}$, where $\epsilon_u = 2^{-53}$ is the unit roundoff.

To compare two phase states, we need a distance that respects periodicity. Near an interval boundary, ordinary subtraction may give nearly a whole period even when the phases are close. We therefore allow a whole-period shift before taking the difference.

Corollary 3.6 (Accumulated circular error)**.** *Under the preceding assumptions, suppose all sums are normal. Let $\widehat{r}_n$ be the computed state after n updates and r_n the exact-arithmetic state with the same initial state, represented increments and period. Then*

$$d_{\widehat{T}}(\widehat{r}_n, r_n) \leq n\epsilon_u \widehat{T}, \qquad d_T(a, b) = \min_{k \in \mathbb{Z}} |a - b - kT|, \tag{3}$$

where d_T is the circular distance for period T.

Proof. Adding the same increment to both states does not change their circular distance. Nor does adding or removing a complete period. The correction is exact by the preceding proposition. Each rounded addition can therefore increase the distance by at most $\epsilon_u \widehat{T}$. The initial distance is zero; summing these bounds over n steps gives the stated inequality. □

With gradual underflow, subnormal terms must be included when a sum is not normal. The bound describes a worst case, not a predicted drift. It does not include errors in constructing the increments, the difference between $\widehat{T}$ and true 2π, or the remainder guard for large steps. NaNs, other rounding modes and transformations that change the stated operations are outside the argument.

Remark 3.7 (Endpoints in other representations)**.** `std::remainder` rounds its quotient to the nearest integer, with ties to even, and can return $+P$ at a half-period. A simple positive update can return $\widehat{T}$ when adding $\widehat{T}$ to a tiny negative subnormal value rounds upward. These values have a valid periodic meaning, although they lie outside the chosen half-open interval. Interval membership and waveform error must therefore be checked separately.

3.3 Fixed point and blockwise sums

The Q48 alternative divides a turn into 2^{48} integer positions. Its state $q/2^{48}$ is measured in cycles. To add a phase increment, we first convert it to this integer scale:

$$q^+ = (q + \mathrm{round}(2^{48} u/\widehat{T})) \bmod 2^{48}. \tag{4}$$

The modulo operation keeps the lowest 48 bits. Unsigned addition and masking are exact. With exact scaling, rounding an increment contributes at most half one phase step. The circular-error bound is therefore $n\widehat{T}/2^{49}$ after n updates. Double-precision scaling introduces further rounding, and sufficiently small increments round to zero. The tested implementation calls `llround` for each changing GFSK increment. Its timing includes that conversion.

The block variants proceed differently. They group $b = 64$ samples and compute each phase offset as the sum of the preceding increments within the block. Such a sum is called a prefix sum. It may span several turns, so the one-correction proposition no longer applies. The experiment reduces the emitted offsets with a floor-based formula and uses remainder for the state passed to the next block.

A serial scan adds increments in sequence. A tree scan groups additions to expose independent work, but requires more operations and storage. Both change the rounding order relative to the scalar oscillator. The two arrays of the tree implementation occupy 1024 bytes of doubles in addition to the waveform buffers.

4 Implementation in WSJT-X

The production generator stores increments and centered phase as doubles. For each sample, it first maps the current phase to a complex amplitude, then advances the phase. Amplitude mapping uses linear interpolation in a complex float table with 65,536 intervals. For ideal sine and cosine values, the interpolation error is bounded by $h^2/8$, where $h = 2\pi/65536$ is the table spacing. This is about $1.15 \cdot 10^{-9}$ per component. Storing the table as floats and rounding during interpolation introduce further errors. Gaussian-pulse construction and the start and end ramps remain specific to each mode.

The measured production source is identified by commit

Table 2: Frame parameters at 48 ksample/s in this fork.

	FT8	FT4	FT2
Modulation tones	8	4	4
Input symbols	79	103	103
Samples/symbol	7680	2304	1152
Output samples	606720	241920	120960
Duration (s)	12.64	5.04	2.52
Gaussian BT	2	1	1

`46ebbd78c663bf07eed054570df4888a6268e156`. All 45 file hashes in the earlier report's manifest match that commit. The research code is identified separately by file hashes, including snapshots of the exact versions used for measurement. Historical manifests retain their original description of the source state at measurement time.

Table 3: Implementation locations; paths are relative to the repository root.

File/function	Role
`lib/ft8/native/ft8_wave.cpp`, `waveform`	Common FT8/FT4/FT2 phase, LUT, pulses, guards and ramps.
Same file, `pulseFor`	At most eight immutable shared pulse objects; one cache lock per frame; scratch belongs to each invocation.
`journal/experiment/phase.hpp`, `oscillate`	Isolated alternative oscillators; generated copies replace only the oscillator segment.
`lib/ft4/native/ft4_sync.cpp`, `search`	Reuses frequency-tweaked Costas references; independent rows may be parallel.
`lib/ft8/native/ft8_ldpc.cpp`	Production native encoder and two BP parity layouts; unchanged in this study.
`lib/ft8/decode174_91.f90`	Standard hybrid BP/OSD, also called by FT4/FT2.

In the table, `journal/experiment` abbreviates `docs/papers/reist-ftx/journal/experiment`. Each experimental copy replaces only the oscillator segment; pulse generation, ramps and output buffers stay the same. The production algorithms are unchanged by the study.

The original Fortran waveform sources remain available for reference tests. They differ in precision, trigonometry and caching, so a timing comparison with them cannot isolate the programming language. Both Fortran and C++ compile to native machine code. FT2 uses the fork's FT4-based integration rather than an independent LDPC code.

5 Experimental methodology

5.1 Timing the phase representations

Measurements use an Intel Core i9-14900K with 24 physical cores and 32 logical CPUs, running Windows 11 Enterprise build 26300. The MSYS2 toolchain provides GCC/GFortran 15.2.0. The waveform programs use C++11 and `-O3 -fno-fast-math -ffp-contract=off`, without LTO or `-march=native`. Only the separate Boost 1.90 decimal-100 check requires C++14; it is not part of the timed waveform kernels.

The processor has two core types. Affinity binds a process to one logical CPU, and CPUID leaf 1A verifies its type [7]: 0x40 for the selected performance core (logical CPU 0), 0x20 for the selected efficiency core (CPU 16). Each selection is measured in three process sessions. Three further sessions allow the operating system to schedule freely. SMT siblings, Turbo, power management and background load are not isolated, so the results describe these selected cores under the recorded conditions.

Each session contains nine blocks per mode. Within a block, all ten implementations run in randomized order after warm-up, generating three complete frames each. They receive the same telemetry word (seed 991), 48 ksample/s and a carrier of 1855 Hz. The word is encoded with the existing LDPC/CRC implementation and the appropriate Costas, Gray and FT4 scrambling mappings. Output consists of complex float IQ samples. Timings include pulse synthesis, allocation, cache access and ramps. As an implementation check, the refactored centered variant must reproduce the production output bytes in the numerical corpus.

The reported speed ratio divides the pooled median time of one variant by that of the centered variant. Uncertainty is assessed by resampling whole process sessions, then paired circular blocks of length three within each selected session (2000 replicates, seed 991202610). Block lengths one and five test sensitivity to that choice. Grouping observations this way preserves some local timing dependence; individual repeated frames are not counted as independent measurements. With only three sessions and shared thermal and operating-system effects, the intervals remain conditional on this experiment. They do not establish population coverage or variation between machines.

Elapsed time is measured with `steady_clock`. As a diagnostic, Windows `QueryThreadCycleTime` supplies accounted user-plus-kernel thread cycles per sample [8]. These are neither retired instruction counts nor calibrated hardware performance-counter readings. Branch misses, cache events, energy and dynamic instruction counts were not collected. Compiler assembly is included for small update loops; masked source expressions alone do not guarantee branch-free machine code.

5.2 Reference runs, numerical checks and spectra

The earlier real-output benchmark is rerun into separate files. It uses the original 21 blocks, ten repeats and seed-991 tone indices. These indices are valid tones but need not encode a message. Its results are kept separate from both the original frozen data and the new experiment with complex output.

The numerical corpus covers three modes, two sample rates (12 and 48 ksample/s), two encoded telemetry messages and five carrier frequencies at each rate: -1855, 0.001, 1855, $0.49 f_s$ and 10^9 Hz. Here f_s denotes the sample rate. This gives 60 frame configurations for each of the ten implementations. The extremely large carrier exercises the general-remainder guard for large steps; it is not an RF operating condition.

The long-double reference accumulates the same double-precision increments using the same represented period, then evaluates long-double sine and cosine. Its significand has 64 bits on this system. It provides a more accurate arithmetic comparison, while retaining the original pulse-construction and period errors. Separate decimal-100 calculations check 4096 local updates, and one million random state/step pairs test interval membership.

Long-sequence tests use one million steps each: constant positive increments, signed random increments and tiny alternating increments. The outputs distinguish circular phase error, maximum and RMS complex IQ error, and exact byte agreement.

For the spectral comparison, a deterministic radix-2 FFT transforms each complete ramped frame with a rectangular observation window and zero-padding to the next power of two. A known-bin test and a Parseval energy check validate the transform. The chosen analysis band extends two tone spacings beyond the nominal lowest and highest tones. It is an analysis convention, not a regulatory emission mask. Error energy is divided by reference-frame energy. All spectra describe digital IQ generated in software.

5.3 Measuring the waveform share of decoder time

A separate executable, excluded from the normal build, times calls to the unchanged production waveform source inside the complete `jt9` file decoder. It processes recorded FT8, FT4 and optional local FT2 samples at depth three with one OpenMP thread. One warm-up run precedes three measured processes; their decoded message sets must agree.

The component share is the sum of the waveform call times divided by the elapsed time from process launch to exit. Startup is therefore included. This profile describes the selected recording files and configuration; it does not measure the GUI, a live stream or a transceiver.

6 Performance results

Table 4: Complex-output waveform time on the selected P-core: median of 27 block means, milliseconds per complete frame. All pulse/ramp work included.

Architecture	FT8 (ms)	FT4 (ms)	FT2 (ms)
Centered	2.774	1.088	0.550
Remainder	7.201	2.857	1.431
Positive	2.860	1.089	0.601
Masked	2.876	1.134	0.557
Q48 DDS	6.482	2.672	1.332
Unbounded	10.857	4.347	2.227
Serial prefix-64	4.758	1.890	1.024
Tree prefix-64	5.864	2.403	1.260
Long double	27.437	10.588	5.386
Production	2.719	1.038	0.528

Table 5: Core dependence of general remainder (G) versus centered (C). Intervals are conditional hierarchical resampling intervals; see methodology.

Core	Mode	G/C	Conditional 95% interval	C (ns/sample)
P	FT8	2.60	[2.35, 2.71]	4.57
P	FT4	2.62	[2.32, 2.75]	4.50
P	FT2	2.60	[2.53, 2.97]	4.55
E	FT8	5.19	[5.05, 5.33]	5.96
E	FT4	4.94	[4.70, 5.08]	6.37
E	FT2	4.43	[4.27, 4.61]	7.04
UNBOUND	FT8	2.63	[2.62, 2.67]	4.16
UNBOUND	FT4	5.03	[2.41, 5.17]	6.46
UNBOUND	FT2	5.53	[1.98, 5.85]	5.62

Positive and centered states take nearly the same time. The main saving comes from avoiding repeated general reduction. Source-level masking does not improve the result consistently. The Q48 version with per-step conversion and the two block scans are slower than scalar centered

correction. These results describe the implementations tested here; they do not rule out faster integer DDS or explicit SIMD designs.

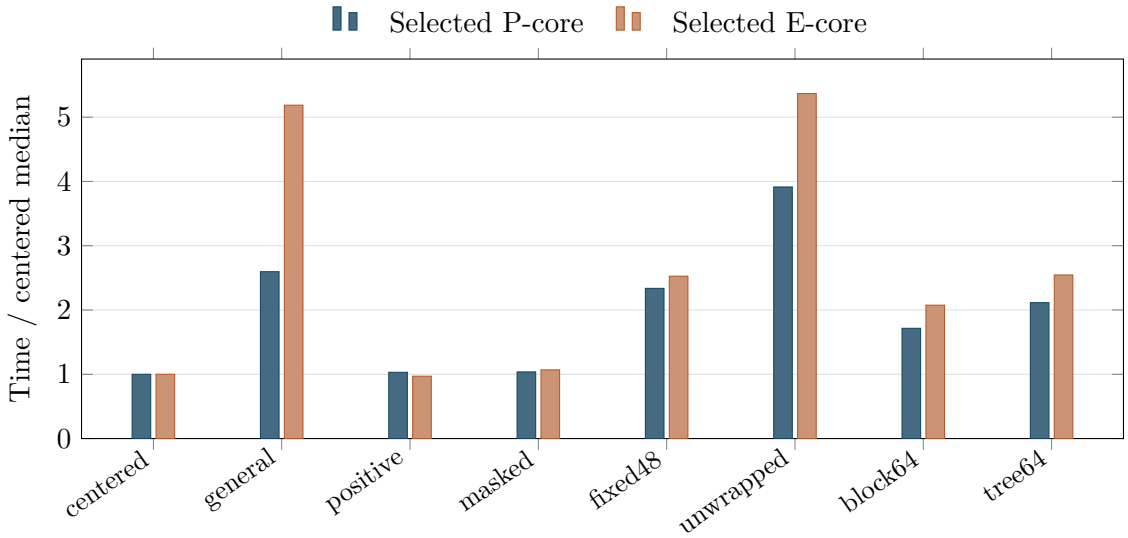

Figure 1: FT8 architecture cost normalized separately within each core class. The figure shows point estimates; interval and sensitivity data are provided in the machine-readable summary.

Table 6: Separate real-output rerun of the historical factorial harness. These ratios should not be conflated with the new complex-output comparison.

Mode	Historical G-L/C-L	New real-output reproduction
FT8	5.55	5.56
FT4	5.24	5.21
FT2	4.99	5.19

The historical rerun checks whether the earlier behavior recurs under the current machine conditions. Exact timings need not match. The P/E and freely scheduled results show why the earlier factors 5.55/5.24/4.99 are not machine-independent constants. Output type and compiler context also differ between the historical and expanded tests.

Separate throughput tests launch independent centered waveform calls with 1, 2, 4 and 8 workers. Each setting uses five blocks and eight frames per worker, including thread creation, scratch allocation and cache locks. These tests allow free scheduling; core-bound sessions are excluded. The raw throughput table is retained. It measures concurrent independent frames, not vectorization of one oscillator, and does not establish how a twelve-worker decoder would scale.

7 Numerical accuracy and signal checks

The centered refactor matches production output byte for byte in all 60 frame configurations. Periodically equivalent alternatives need not have identical bytes. After one million constant positive steps, centered correction and general remainder reach a maximum circular error of about $6.86 \cdot 10^{-11}$ rad relative to the long-double reference with the same represented period. The positive state reaches $1.53 \cdot 10^{-10}$ rad, Q48 about $2.38 \cdot 10^{-9}$ rad, and the unbounded accumulator about $4.40 \cdot 10^{-6}$ rad.

Table 7: Worst result across the 60-frame corpus versus the long-double reference. IQ maximum and RMS use complex magnitude; phase excludes ramps whose reference amplitude is at most 0.1.

Architecture	Maximum IQ error	Maximum RMS	Maximum phase error (rad)
Centered	8.43e-08	3.20e-08	1.30e-07
Remainder	8.43e-08	3.20e-08	1.30e-07
Positive	8.43e-08	3.20e-08	1.30e-07
Masked	8.43e-08	3.20e-08	1.30e-07
Q48 DDS	8.43e-08	3.20e-08	1.30e-07
Unbounded	2.22e-06	8.80e-07	2.27e-06
Serial prefix-64	8.43e-08	3.20e-08	1.30e-07
Tree prefix-64	8.43e-08	3.20e-08	1.30e-07

Those values belong to one step sequence. Signed random increments behave differently, so the numbers are not general drift rates. Tiny positive steps can disappear entirely in Q48. The boundary tests also reproduce the remainder tie and positive-state endpoint cases explained in the mathematical section.

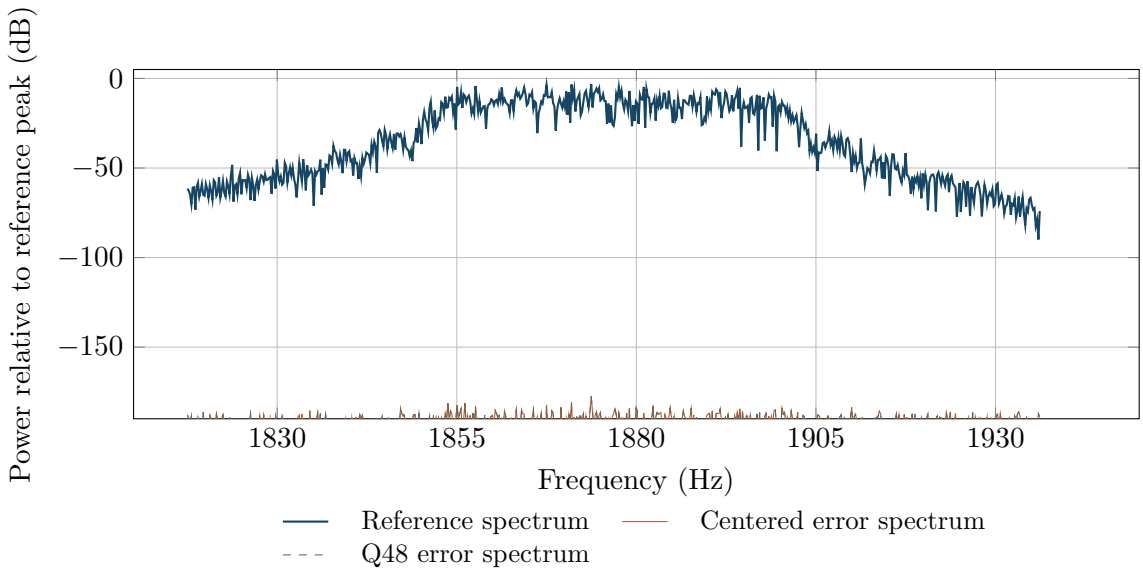

Figure 2: One generated FT8 telemetry frame at 1855 Hz and 48 ksample/s. Error spectra use IQ differences, not the difference between logarithmic power curves. This is a software comparison, not a transmitter measurement.

For the three selected frames, the integrated energy of the difference between centered output and reference is about −150 dBc. Q48 gives a similar result at this level of float-output error. Relative Parseval errors are below $7 \cdot 10^{-13}$. Reference energy outside the chosen analysis band is about −38.1, −48.3 and −50.9 dBc for FT8, FT4 and FT2. These values depend on the observation window, ramps and band limits; they neither certify an emission mask nor show reduced RF sidelobes. The reference and variants still share the double pulse increments, final float conversion and ramp coefficients.

Table 8: Unchanged waveform component inside the complete single-thread recording decoder; medians of three runs after warm-up. The ideal bound removes this component entirely while holding all other work constant.

Mode	Wall (ms)	Wave (ms)	Calls	Share (%)	Ideal bound
FT8	1574.7	33.47	254	2.13	1.022
FT4	177.7	6.63	18	3.73	1.039
FT2	144.6	3.46	5	2.39	1.024

8 Practical significance for the complete decoder

Waveform generation accounts for 2.13% of elapsed FT8 decoder time, 3.73% for FT4 and 2.39% for FT2 in these runs. This small share limits what further waveform optimization can achieve. Removing it entirely would reduce time by those percentages. The corresponding speed increases would be 2.17%, 3.87% and 2.45%; time saved and speed gained use different denominators.

We now describe this relationship in general, using Amdahl's model. Let p be the component's share of the baseline time and S its speedup factor. Normalize total baseline time to one. The unchanged work then takes $1 - p$, and the accelerated component takes p/S. Total speedup is the reciprocal of their sum:

$$S_{\text{total}} = \frac{1}{1 - p + p/S}, \qquad S_{\text{total}} \leq \frac{1}{1 - p}. \tag{5}$$

For the upper limit, assume that the component takes no time at all. Only the remaining work then contributes to elapsed time. It applies to the already optimized generator, these recordings and the measured single-thread configuration. It does not bound the cumulative gain of earlier changes or predict other workloads.

The same model can describe a hypothetical reversal of the optimization. If general remainder costs a factor G more than the current centered component, total time would rise by the factor $1 + (G - 1)p$, assuming the rest stayed constant. No exact receiver value is claimed for this factor: sample rate, cache state and input mix can change G. In particular, the 48 ksample/s transmit-waveform ratio cannot simply be substituted for a measured receive-path ratio.

9 Discussion

The comparison favors bounded state updates for this GFSK implementation. It also shows that a positive interval can provide essentially the same speed as a centered one. Centering is useful for signed phase errors, and its corrective subtraction is exact under the stated binary64 conditions. The size of its timing advantage over library remainder depends strongly on core type and compilation context.

The slower alternatives point to work that remains. Q48 replaces state rounding with increment quantization and conversion cost; a conversion path more efficient than per-step `llround` might change its result. Similarly, the tree scan compiled for the default instruction set does not settle the value of SIMD. Explicit AVX2 scans, combined pulse/phase representations, vector conversion and batches of independent oscillators are candidates for further tests. Each change to addition order would need a fresh error analysis.

For the measured complete decoder, however, the waveform profile sets a low ceiling on further gains from this component alone. Research aimed at decoding more messages should also examine synchronization and error correction. The separate decoder dossier treats those questions with their own measures of success, cost and false acceptance.

10 Limits of the evidence

The timing evidence comes from one physical machine and one compiler version. There are no AMD or ARM64 measurements, hardware-counter or energy results, or RF measurements. Core types were verified, but power and background activity were not controlled. Three process sessions provide limited evidence about scheduler and temperature effects. Resampling intervals can therefore miss systematic uncertainty. The ratio of pooled medians also differs from the median of paired ratios.

The historical and expanded experiments have different output types, inputs and compilation contexts. Both include pulse synthesis and ramps, so their timings describe a complete waveform kernel, not one isolated phase addition. The numerical reference shares the double increments and represented period. Decimal-100 checks of local updates do not validate the entire Gaussian pulse to 100 digits. Two telemetry messages per setting and a finite test corpus cannot prove byte identity for all possible inputs. The spectra cover selected generated frames; no audio interface, USB path, amplifier, antenna or FT-991A was measured.

A runner is provided for replication on a real AMD or ARM64 host. It uses new output directories and records source hashes, toolchain and execution metadata for the same waveform workload. That replication is still pending. An x86 virtual machine or emulated ARM process on this i9 would not supply a second physical platform. Long-double precision must also be recorded for each ABI; its 64-bit significand here is not a universal property.

External replication, independent mathematical review and journal peer review remain outstanding.

11 Conclusions

If phase and increments are bounded, one correction can keep the oscillator state within a single period. The real-valued argument carries over to the specified binary64 update because the corrective subtraction is exact. Rounding in the preceding addition and in waveform mapping still has to be accounted for.

On the measured i9-14900K, this update is substantially faster than general remainder. A positive state performs about as well as a centered state, and the measured ratios differ between performance and efficiency cores. REIST motivates the centered-state formulation; the empirical contribution is the controlled comparison and its numerical checks. The application profile gives the practical scale: further optimization of the waveform component alone can increase processing speed by at most about 2.2–3.9% for these file-decoder workloads. Larger improvements in reception require separate evidence from the rest of the receiver.

12 Data and code availability

The application source is available at `https://github.com/rudolfstepan/wsjtx`. The measured production base is commit `46ebbd78c663bf07eed054570df4888a6268e156`; the research sources and version 0.2 of this study were published in commit

`3f1a297ed05e4351553ee50c33934c1cd3d0675b`. The accompanying package identifies the current documents and the exact measured source versions by hash. This version 0.3 revises the exposition and adds no measurements.

Historical raw data remain in `docs/papers/reist-ftx/data`; the expanded study uses `docs/papers/reist-ftx/journal/results`. Versioned Python scripts generate tables and plots

without pooling the two datasets. The supplied original REIST PDF and optional third-party FT2 recordings are not redistributed.

`Build-Study.ps1` runs the documented build, analysis and LaTeX workflow and can create new measurement directories. Standalone waveform experiments require GNU C++, Boost headers, CMake, Ninja and Python; historical reproduction also requires Fortran. Complete decoder and profile tests need the full fork and its usual dependencies. The source retains GPLv3 and upstream credits. A publication license for the draft has not been chosen, and no journal submission or new DOI registration has been made.

Author contributions and AI assistance

Rudolf Stepan supplied the REIST formulation, existing source fork, research objectives and original manuscript. OpenAI Codex assisted with the source audit, experimental implementation, local execution, analysis, figures and substantial drafting, including this language revision. Scientific verification and responsibility for any submission remain with the author; journal-specific AI policies must be checked separately. The original WSJT-X protocol and software developers and the fork's documented FT2 contributors retain their authorship and license credits.

References

[1] R. Stepan, *REIST-Division: Eine implementierungsorientierte Formulierung zentrierter Restarithmetik. Theorie, Architekturanalyse und empirische Evaluation auf ARMv8-A, x86-64 und FPGA*, supplied manuscript, 2026. DOI as printed: 10.5281/zenodo.21206471; public metadata not independently retrieved. Original PDF SHA256 in the audit.

[2] R. Stepan, *REIST Division als Zustandsprinzip in den GFSK Rechenkernen von FT8 FT4 und FT2*, technical report v1.0, 15 October 2026, `docs/papers/reist-ftx/paper.tex`, archived with commit 46ebbd78c.

[3] Analog Devices, *MT-085: Fundamentals of Direct Digital Synthesis (DDS)*, technical tutorial. `https://www.analog.com/media/en/training-seminars/tutorials/mt-085.pdf`.

[4] D. Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, vol.23, no.1, pp.5–48, 1991. Authoritative reprint: `https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html`.

[5] G. E. Blelloch, *Prefix Sums and Their Applications*, Carnegie Mellon University, CMU-CS-90-190, November 1990. `https://www.cs.cmu.edu/afs/cs.cmu.edu/project/scandal/public/papers/CMU-CS-90-190.html`.

[6] S. Franke, B. Somerville and J. Taylor, "The FT4 and FT8 Communication Protocols," *QEX*, July/August 2020, pp.7–17. `https://wsjt.sourceforge.io/FT4_FT8_QEX.pdf`.

[7] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual*, CPUID leaf 1AH, hybrid core-type enumeration. `https://cdrdv2-public.intel.com/868137/325462-089-sdm-vol-1-2abcd-3abcd-4.pdf`.

[8] Microsoft, *QueryThreadCycleTime function*, Win32 API documentation. `https://learn.microsoft.com/en-us/windows/win32/api/realtimeapiset/nf-realtimeapiset-querythreadcycletime`.

[9] L. Cordesses, "Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 1)," *IEEE Signal Processing Magazine*, vol.21, no.4, pp.50–54, 2004. DOI: 10.1109/MSP.2004.1311140.

[10] J. Vankka, *Direct Digital Synthesizers: Theory, Design and Applications*, doctoral dissertation, Helsinki University of Technology, November 2000, Chapters 5–6 and 9. https://aaltodoc.aalto.fi/server/api/core/bitstreams/82529cd1-8a15-4773-bbc3-18a27ce66376/content.

[11] J. Vankka, "Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis," *1996 IEEE Frequency Control Symposium*, pp.942–950, 1996. https://research.aalto.fi/en/publications/methods-of-mapping-from-phase-to-sine-amplitude-in-direct-digital/.

[12] J. Vankka, "Spur Reduction Techniques in Sine Output Direct Digital Synthesis," *1996 IEEE Frequency Control Symposium*, pp.951–959, 1996. https://research.aalto.fi/en/publications/spur-reduction-techniques-in-sine-output-direct-digital-synthesis/.

[13] K. I. Palomäki and J. Nurmi, "Taylor Series Interpolation-Based Direct Digital Frequency Synthesizer with High Memory Compression Ratio," *Sensors*, vol.25, no.8, article 2403, 2025. DOI: 10.3390/s25082403.

[14] M. Vicanek, *A New Recursive Quadrature Oscillator*, author-published technical note, 21 October 2015. https://vicanek.de/articles/QuadOsc.pdf.