From Finite-Field Polynomials to CCSDS Reed–Solomon in a Practical Telemetry Chain

1 Finite Fields & Modular Arithmetic (the algebra RS uses)

Reed–Solomon (RS) codes live over a finite field GF(q). For space links and CCSDS, $q = 2^8 = 256$ so one *symbol* is one byte. All arithmetic is performed *in the field*:

- In GF(256): addition/subtraction is bitwise XOR; multiplication/division are polynomial operations modulo a fixed primitive (irreducible) degree-8 polynomial (commonly $x^8 + x^4 + x^3 + x^2 + 1$).
- In toy examples we often use GF(7) (integers mod 7) so that inverses and products are computed "mod 7".

2 Message as a Polynomial; Encoding by Evaluation

Given k symbols $m_0, \ldots, m_{k-1} \in \mathrm{GF}(q)$, define the polynomial

$$m(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}. \tag{1}$$

Choose n distinct field points $a_1, \ldots, a_n \in \mathrm{GF}(q)$ and *encode* by evaluating

$$c_i = m(a_i), \qquad i = 1, \ldots, n. \tag{2}$$

The vector $(c_1, \ldots, c_n)$ is the RS codeword of an (n, k) code. RS codes are *MDS*:

$$d_{\min} = n - k + 1, \quad \text{correct up to } t = \left\lfloor \frac{n-k}{2} \right\rfloor \text{ errors, or any } e \text{ erasures with } 2t + e \le n - k.$$

3 Interpolation & Lagrange Polynomials (why recovery works)

Any degree-$< k$ polynomial is uniquely determined by any k samples at distinct points. The explicit reconstruction is Lagrange interpolation:

$$m(x) = \sum_{i=1}^{k} y_i \, \ell_i(x), \qquad \ell_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j}, \tag{3}$$

with all operations in GF(q). For *erasures* (positions known), this solves the whole decode. For *errors* (positions unknown), classic RS decoders compute syndromes, then solve the key equation to find the error-locator $\Lambda(x)$ (e.g., via Berlekamp–Massey or Euclidean algorithm), locate roots (Chien search), and compute magnitudes (Forney's formula).

4 Mini Worked Example over GF(7) ("mod 7")

Let $k = 3$, $n = 6$, and $m(x) = 3 + 2x + 4x^2 \pmod 7$. Transmit $c_i = m(i)$ for $i = 1, \ldots, 6$. Suppose the channel erases positions $i = 4$ and corrupts one position. If we still have any $k = 3$ correct pairs (x_i, y_i), Lagrange interpolation in GF(7) recovers $m(x)$ exactly; re-evaluating gives every c_i, filling the erasure and correcting the wrong symbol. This is the intuition behind RS erasure and error correction. (*Note:* with unknown error positions, use the syndrome/locator route, then interpolate the cleaned set.)

5 Definition: Interleave Depth I

In a CCSDS RS interleaver of depth I, we form I RS codewords *in parallel* and transmit their bytes in a round-robin (column-wise) order *on the same physical channel*. If j counts transmitted RS-coded bytes (after the ASM), then

$$\text{row} = j \bmod I, \qquad \text{column} = \left\lfloor \frac{j}{I} \right\rfloor.$$

At the receiver, the de-interleaver inverts this mapping to reassemble each 255-byte RS word. A time-contiguous burst of length B bytes is spread to approximately B/I byte errors per RS row.

6 Your CCSDS Chain with Numbers (as in your GRC)

We consider one TM transfer frame of **1115** bytes, randomized, then RS(255,223) with depth I=5, the 4-byte ASM `0x1ACFFC1D`, convolutional code r=1/2, K=7 with 6 tail bits, and OQPSK modulation.

Stage-by-stage sizes

- TM input: 1115 B = 5×223 (fits exactly into $I = 5$ rows).
- RS(255,223): add 32 B parity per row $\Rightarrow$ total parity $5 \times 32 = 160$ B.
- Post-RS (interleaved): $1115 + 160 = 1275$ B.
- CADU (add ASM 4 B): 1279 B.
- Pre-convolution bits: $1279 \times 8 = 10232$ bits.
- Post-convolution (rate-1/2 with 6 input tail bits): $2 \times (10232 + 6) = \mathbf{20476}$ bits.
- OQPSK symbols (2 bits/sym): $20476/2 = \mathbf{10238}$ symbols.

Burst tolerance intuition with I=5

Each RS row can correct up to 16 byte errors (or 32 erasures). A contiguous channel burst of B bytes is de-interleaved to roughly B/I per row. Thus $B \lesssim 16I = \mathbf{80}$ bytes is generally recoverable.

Concept / Quantity	Equation / Definition
Finite field	All ops in GF(q); for CCSDS q=256, add = XOR, mult/div mod primitive poly.
Message polynomial	$m(x) = \sum_{i=0}^{k-1} m_i x^i$ over GF(q)
Encoding (evaluation view)	$c_i = m(a_i)$ at n distinct $a_i \in \mathrm{GF}(q)$
Minimum distance & capability	$d_{\min} = n - k + 1$, errors $t = \left\lfloor \frac{n-k}{2} \right\rfloor$, erasures e with $2t + e \le n - k$
Lagrange interpolation	$m(x) = \sum_{i=1}^{k} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$
Syndromes	$S_\ell = \sum_{i=1}^{n} r_i a_i^\ell, \quad \ell = 0, \ldots, n-k-1$
Key equation	$\Lambda(x)S(x) \equiv \Omega(x) \pmod{x^{n-k}}$
Error locations	Roots of $\Lambda(x)$ via Chien search
Error magnitudes	Forney: $E(x_i) = -\frac{\Omega(x_i)}{\Lambda'(x_i)}$
Interleave mapping	row $= j \bmod I$, col $= \lfloor j/I \rfloor$ (after ASM)
CADU size (yours)	1279 B = 4 B ASM + (5 × 255 B) with 5 × 223 info and 5 × 32 parity
Conv. output bits	$N_{\text{out}} = \frac{1}{r}(N_{\text{in}} + \text{tail})$; here $2 \times (10232 + 6) = 20476$
Symbols (OQPSK)	$N_{\text{sym}} = N_{\text{out}}/2 = 10238$
Throughput fraction	$\eta \approx \frac{8 \times 1115}{20476} \approx 0.436$ (payload bits / channel bits)

7 Key Equations Table

8 Receiver Sketch (aligning with your GRC blocks)

1. **Timing/Carrier**: Pluto (or other) → LPF → Polyphase Clock Sync (timing) → OQPSK de-stagger → Costas (carrier/phase).
2. **Soft bits**: Constellation soft decoder (LLRs), optional sign flip.
3. **Bit FEC**: Viterbi (rate 1/2, $K = 7$) → hard bits.
4. **Frame lock**: ASM correlate (`0x1ACFFC1D`) to find CADU boundaries.
5. **Byte FEC**: De-interleave (I=5) → RS(255,223) decode (up to 16 byte errors/row).
6. **Source data**: Derandomize (PRBS XOR) → original 1115 B TM frame → KISS/app.

9 Takeaways

RS codes are "sample the polynomial" codes over a finite field. The *algebra* (finite-field modular arithmetic) lets you evaluate, invert, and divide; the *geometry* (polynomial interpolation) guarantees any k good samples recover the message. In practice (your CCSDS chain), RS protects bytes (with interleaving to tame bursts), the convolutional code protects bits (with soft Viterbi), and the ASM is your robust frame marker.