```
// src/codec/frameEncoder.ts

import { WnspFrame } from "../protocol/frameTypes";
import { encodeTextToWavelengthSequence } from "./
textEncoder";
import { getWavelengthForLetter } from "../protocol/
wavelengthMap";

/**
 * Default sync pattern for v1.0.
 */
export const DEFAULT_SYNC_PATTERN = 0xaa;

/**
 * Default intensity level for v1.0 (0–7).
 */
export const DEFAULT_INTENSITY_LEVEL = 4;

/**
 * Compute a simple checksum by XOR-ing the core numeric
fields.
 * This is intentionally simple; future versions may use
stronger methods.
 */
export function computeChecksum(
  sync: number,
  wavelengthNm: number,
  intensityLevel: number,
  payloadBit: 0 | 1
): number {
```

```
  let checksum = 0;
  checksum ^= sync & 0xff;
  checksum ^= wavelengthNm & 0xff;
  checksum ^= intensityLevel & 0xff;
  checksum ^= payloadBit & 0xff;
  return checksum;
}

/**
 * Encode a sequence of wavelengths into WNSP frames.
 * In v1.0:
 * - payloadBit is derived as parity of the letter index (if
provided)
 *   or 0 if unknown.
 */
export function encodeWavelengthsToFrames(
  wavelengths: number[],
  options?: {
    intensityLevel?: number;
    syncPattern?: number;
  }
): WnspFrame[] {
  const sync = options?.syncPattern ??
DEFAULT_SYNC_PATTERN;
  const intensity = options?.intensityLevel ??
DEFAULT_INTENSITY_LEVEL;
  const frames: WnspFrame[] = [];

  const nowBase = Date.now();
```

```
  for (let i = 0; i < wavelengths.length; i++) {
    const wl = wavelengths[i];
    const parity = (i % 2) as 0 | 1;
    const checksum = computeChecksum(sync, wl, intensity,
parity);

    const frame: WnspFrame = {
      sync,
      wavelengthNm: wl,
      intensityLevel: intensity,
      payloadBit: parity,
      checksum,
      timestampMs: nowBase + i,
    };

    frames.push(frame);
  }

  return frames;
}

/**
 * Convenience encoder: text string -> frames.
 */
export function encodeTextToFrames(
  text: string,
  options?: {
    intensityLevel?: number;
    syncPattern?: number;
  }
```

```
): WnspFrame[] {
  const wavelengths =
encodeTextToWavelengthSequence(text);
  return encodeWavelengthsToFrames(wavelengths, options);
}

/**
 * Utility to get a simple "index parity" for a letter based on its
position in alphabet.
 */
export function getLetterParity(letter: string): 0 | 1 {
  const upper = letter.toUpperCase();
  if (upper < "A" || upper > "Z") return 0;
  const index = upper.charCodeAt(0) - "A".charCodeAt(0);
  return (index % 2) as 0 | 1;
}

/**
 * Encode letters directly to frames (if you already have
letters).
 */
export function encodeLettersToFrames(
  letters: string[],
  options?: {
    intensityLevel?: number;
    syncPattern?: number;
  }
): WnspFrame[] {
  const sync = options?.syncPattern ??
DEFAULT_SYNC_PATTERN;
```

```
  const intensity = options?.intensityLevel ??
DEFAULT_INTENSITY_LEVEL;
  const frames: WnspFrame[] = [];
  const nowBase = Date.now();

  for (let i = 0; i < letters.length; i++) {
    const letter = letters[i];
    const wl = getWavelengthForLetter(letter);
    if (typeof wl !== "number") continue;

    const parity = getLetterParity(letter);
    const checksum = computeChecksum(sync, wl, intensity,
parity);

    frames.push({
      sync,
      wavelengthNm: wl,
      intensityLevel: intensity,
      payloadBit: parity,
      checksum,
      timestampMs: nowBase + i,
    });
  }

  return frames;
}
```