UK postcode

UK postcode in either compact or spaced form.

```
^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`[A-Z]{1,2}`	between 1 and 2 times: any of: uppercase letters
`\d`	any digit (0–9)
`[A-Z\d]?`	optional (zero or one): any of: uppercase letters, digits (0–9)
`\s?`	optional (zero or one): whitespace
`\d`	any digit (0–9)
`[A-Z]{2}`	exactly 2 times: any of: uppercase letters
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$/;
re.test(value);
```

Python

```
import re
re.match(r"^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$", value)
```

Java

```
Pattern.compile("^[A-Z]{1,2}\\d[A-Z\\d]?\\s?\\d[A-Z]{2}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$");
```

Go

```
regexp.MustCompile(`^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$`).MatchString(value)
```

Ruby

```
/^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$/.match?(value)
```

PHP

```
preg_match('~^[A-Z]{1,2}\d[A-Z\d]?\s?\d[A-Z]{2}$~', $value);
```

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and $, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Uppercase only.** Ranges like [A-Z] won't match lowercase. Add a-z (or the i flag) if lowercase input should be accepted.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the uk postcode against a source of truth (database, API, or checksum) where it matters.