Module	Inputs (What it receives)	Outputs (What it produces)	Called By (From where)	Calls Into (To where)	Notes / Rules
driver (main / orchestrator)	CLI args, input filename, compile-time flags (OUTFORMAT, DEBUG, COUNTCONFIG)	exit code, initialized modules, (future) `token_list`	OS / user execution	`char_stream`, `logger`, `counter`, `token_list`, `automata`, `out_writer`, `error`	Wrapper only. No scanning, no formatting.
lang_spec (language definition)	`char ch`	character class results, keyword tables, category strings	`automata`, `out_writer`	—	Single source of language constants. No dependencies.
char_stream (input cursor)	`FILE*`, internal cursor state	`cs_peek()`, `cs_get()`, line/col	`driver` initializes; **automata uses**	OS file I/O	🟢 **Cursor only**. **Never classifies, skips, or groups characters.**
automata (scanner engine)	characters via `cs_peek/cs_get`, `lang_spec` rules	tokens, error events, count events	`driver`	`char_stream`, `lang_spec`, `token`, `token_list`, `error`, `counter`, `logger`	🟢 **ONLY module allowed to consume input (`cs_get`).** All lexical decisions live here.
token (token object + enum)	lexeme buffer, category, line	`token_t`	`automata`, `token_list`, `out_writer`	—	Pure data object. No I/O.
token_list (ordered token stream)	`token_t`	ordered token list	`driver`, `automata`, `out_writer`, (future) parser	`token` (free)	Stores tokens only. No formatting or scanning logic.
out_writer (`.cscn` formatting)	tokens grouped by line, output mode	formatted `.cscn` file	`driver`	`token`, `lang_spec`, `logger`	RELEASE/DEBUG formatting only. No scanning.

logger (debug router)	messages, config flags	routed output (stdout/file/dbgcnt)	`driver`, `automata`, `error`, `counter`, `out_writer`	stdout / files	Centralized routing via `fprintf`.
error (catalog + reporter)	error ID, step ID, line, context	formatted error message	`automata`, `driver`	`logger`	Centralized error handling.
counter (COUNTCONFIG)	increment events, context	per-function & global totals	`driver`, `automata`	`logger`	Compiles out when disabled.
parser (future hook)	token list or `.cscn`	parse structures	later project phase	`token_list`	Planned extension (dashed dependency).