Perfect — let's document `_extract_connect_blocks`, since this is step **3 of `parse_spd`**. This is where the parser finds the **IC port connections** and **decap port connections** inside the `.spd`.

📘 Documentation

`_extract_connect_blocks(text: str) -> Tuple[List[str], List[str]]`

Purpose

Extracts `.Connect … .EndC` blocks from the SPD file that describe:

- **IC ports** (where the chip connects to the PDN).
- **Decap ports** (where decoupling capacitors connect).

Returns them as two separate lists, while preserving the order they appear in the file.

How it works

1. **Locate the component section**
    - Many SPD files have a section that begins with:

      ```
      luaCopy code* Component description lines
      ```

    - The function tries to capture everything inside this section, up until the next `*` line.
    - If that section isn't found, it falls back to scanning the **entire file**.

```
pythonCopy codecomp_match = re.search(r"\*\s*Component
description lines\b(.*?)(?:\n\*)", ...)
component_block = comp_match.group(1) if comp_match else text
```

---

2. **Extract IC port blocks**
    - Matches `.Connect ic_port ...` through `.EndC`.
    - Example SPD snippet:

```
rubyCopy code.Connect ic_port1
Node10::GND+
Node12::PWR+
.EndC
```

- Regex:

```
regexCopy code\.Connect\s+ic_port[^\n]*\n(.*?)(?:\.EndC\b)
```

- Captures only the **inner body** (the lines between `.Connect` and `.EndC`).

→ Returns a list like:

```
pythonCopy codeic_blocks = [
  "Node10::GND+\nNode12::PWR+",
  "Node11::GND+\nNode13::PWR+"
]
```

---

3. **Extract decap/cap port blocks**
   - Matches `.Connect decap_portX` or `.Connect cap_portX` through `.EndC`.
   - Example SPD snippet:

```
rubyCopy code.Connect decap_port1
Node14::GND+
Node15::PWR+
.EndC
```

   - Regex:

```
regexCopy code\.Connect\s+(?:decap|cap)_port\d*\s+[^\n]*\
n(.*?)(?:\.EndC\b)
```

   - Also returns only the **inner body**.

   → Returns a list like:

```
pythonCopy codedecap_blocks = [
  "Node14::GND+\nNode15::PWR+"
]
```

---

4. **Return value**

```
pythonCopy codereturn ic_blocks, decap_blocks
```

- `ic_blocks`: list of strings, each the body of an IC connection block.
- `decap_blocks`: list of strings, each the body of a decap connection block.

Example (from `b4_1.spd`)

Input SPD excerpt:

```
rubyCopy code* Component description lines
.Connect ic_port1
Node10::GND+
Node12::PWR+
.EndC

.Connect decap_port1
Node14::GND+
Node16::PWR+
.EndC
```

Parsed result:

```
pythonCopy codeic_blocks = [
  "Node10::GND+\nNode12::PWR+"
]

decap_blocks = [
  "Node14::GND+\nNode16::PWR+"
]
```

Why it matters

- These `.Connect` blocks define **which nodes are grouped into a port**.
- IC ports → define where the chip attaches to the PDN.
- Decap ports → define where capacitors are soldered.
- Later, `_fill_ic_decap_vias` uses these blocks + `node_info` (from step 2) to build `brd.ic_via_xy`, `brd.decap_via_xy`, etc.

 Summary:
`_extract_connect_blocks` finds `.Connect` blocks in the SPD file, splits them into **IC vs decap**, and returns their bodies. This provides the raw text that later steps convert into via coordinates and port groupings.

Do you want me to next document **`_fill_ic_decap_vias`**, which consumes these blocks and actually creates the via coordinate arrays (`brd.ic_via_xy`, `brd.decap_via_xy`)?