🚨 The Problem: `sscanf` Fails with `%f` on ARM (STM32 HAL)

In many **STM32 embedded projects** (especially when using **GCC** and **newlib C library**), functions like:

```
float value;
sscanf("+DATA:1,123.45", "+DATA:%*d,%f", &value);
```

will **not correctly parse the float value**.
The float (`%f`) is either ignored, returns 0, or gives garbage values.

⚙ Why This Happens:

1. **Embedded Systems are Resource-Constrained:**
   STM32 microcontrollers often have **limited flash memory and RAM**.

2. **newlib (the standard C library for ARM) disables floating-point I/O by default:**
   To **save space**, newlib comes with floating-point support for functions like `printf`, `scanf`, `sscanf` **disabled**.

3. `%f` in `sscanf` is considered **heavy** because:

   - It involves converting ASCII text (e.g., `"123.45"`) into an actual floating-point number (`float`).
   - This needs additional code for parsing decimals, exponentials, etc.

📉 Consequences:

- `sscanf` with `%f` → **fails silently** (no compile-time error, but incorrect runtime behavior).
- Many STM32 developers waste time debugging this, not realizing it's a **linker configuration issue, not a code issue**.

✅ The Solution: Enable Floating-Point Parsing in newlib (`_scanf_float`)

What is `_scanf_float`?

- `_scanf_float` is a **symbol** in newlib that, when **forced into the final linked binary, includes the floating-point parsing code**.

🔗 How to Enable It (Step by Step):

🛠 If you use **STM32CubeIDE** (most common case):

1. Right-click your project → **Properties**.
2. Go to:

```
C/C++ Build → Settings → Tool Settings → MCU GCC Linker →
Miscellaneous
```

3. In the **Other Flags** box, add:

```
-u _scanf_float
```

4. Apply → Clean → Build.

🛠 If you use **Makefile**:

In your `Makefile`, add to `LDFLAGS`:

```
LDFLAGS += -u _scanf_float
```

✅ Optional: Enable Floating-Point `printf` too (if needed):

PROF

- Add:

```
-u _printf_float
```

*(only if you need to print floats using `printf("%f", value);` in embedded code—this is **separate** from `scanf`)*

📊 Memory Considerations:

- Enabling `_scanf_float` adds about **1–2 KB** to flash memory usage (depends on compiler and optimization).

- If you need only **parsing floats** (not printing), you can enable just `_scanf_float` and skip `_printf_float`.

📝 Practical Example:

Before:

```
float value;
sscanf("+DATA:1,123.45", "+DATA:%*d,%f", &value);
printf("Value: %f\n", value);  // value will be zero or garbage
```

After adding `-u _scanf_float`:

```
float value;
sscanf("+DATA:1,123.45", "+DATA:%*d,%f", &value);
printf("Value: %f\n", value);  // works correctly
```

🔑 Summary:

Issue	Cause	Fix
`sscanf` with `%f` fails	Floating-point parsing is stripped	Add `-u _scanf_float` linker flag
`printf` with `%f` prints nothing	Floating-point printing is stripped	Add `-u _printf_float` linker flag