🚀 PowerMill Community Macro Assistant

Setup & Contribution Guide

This guide explains how to access, build, and contribute to the **PowerMill Community Macro Assistant** — a flexible AI assistant designed to help generate, debug, and improve Autodesk PowerMill macros.

✅ Requirements

To access the GPT Builder, you must have:

- ChatGPT **Plus**
- ChatGPT **Team**
- ChatGPT **Enterprise**
- ChatGPT **Pro**

⚠️ The GPT Builder is not available on the Free plan.

🔧 Option 1 — Use the Official Community GPT (Recommended)

If an official link is provided:

1. Click the shared GPT link.
2. Log in.
3. Start generating macros.
4. Test in PowerMill.
5. Report improvements or share better approaches.

This is the easiest way to participate.

🏗️ Option 2 — Build Your Own Community Copy (Advanced Users)

If you want to experiment or help improve architecture:

Step 1 — Open the GPT Builder

1. Go to: https://chat.openai.com
2. Log in
3. Click **Explore GPTs** (left sidebar)
4. Click **Create** (top right)

Step 2 — Configure the GPT

Name

PowerMill Macro Assistant

Description

A flexible Autodesk PowerMill macro assistant that generates, debugs, and improves macro automation using pattern-driven and parameter-based architectures.

Step 3 — Paste the Instructions

Copy the official Community Instructions block provided in the project documentation and paste it into the **Instructions** section.

This defines:

- camelCase naming standard
- pattern ↔ handler 1:1 mapping
- logging and error handling rules
- switchboard compatibility
- parameter handling philosophy
- flexible architecture support

Step 4 — (Optional) Upload Example Macros

If you want it to understand your shop's structure:

- Upload:
    - Switchboard macro

    - Function library macro
    - Example toolpath macro
    - Parameter file (.pmlpar)

⚠️ Only upload files you are allowed to share.

Step 5 — Save

Click **Save** and choose visibility:

- **Only Me** → Private version
- **Anyone with link** → Shareable
- **Public** → Discoverable by all ChatGPT users

🧠 How to Contribute to the Community

The assistant improves through structured feedback.

Best Ways to Help

1️⃣ Share Macro Patterns

Provide examples of:

- Different switchboard styles
- Tool creation flows
- Feature-based automation
- NC output handling
- Error logging strategies

2️⃣ Submit Improvements

If the assistant:

- Generates inefficient logic
- Misses preflight checks
- Doesn’t match best practice

- Uses fragile logic

Refine it and share the improved version.

3️⃣ Standardize Best Practices

Help evolve:

- Trigger naming conventions
- Logging standards
- Error mitigation approaches
- Parameter architecture

📐 Community Baseline Standards

These are the shared rules across contributors:

Naming

- camelCase for new patterns, functions, locals

Trigger Architecture

- Default: pattern name matches handler name
    - CREATE PATTERN "lifterStage1"
    - routes to lifterStage1()

Logging

- MESSAGE INFO for debug
- Optional persistent error log file

Error Handling

- Preflight checks before risky commands
- Graceful exit when possible
- Log stage before executing high-risk operations

🛠 Recommended Testing Workflow

1. Generate macro
2. Read through it
3. Add MESSAGE INFO debug lines if needed
4. Test in PowerMill
5. Identify improvements
6. Refine prompt or update GPT instructions

🌎 Community Goal

The objective is not just to write macros faster.

It is to:

- Increase macro adoption in the CAM industry
- Standardize clean automation patterns
- Make advanced automation accessible
- Build a shared macro knowledge base
- Reduce fragile, undocumented scripting

🔐 Data Awareness

When uploading files:

- Do not upload proprietary or restricted IP unless allowed
- Public GPTs share uploaded “Knowledge” with users of that GPT

🔄 Long-Term Vision

This can evolve into:

- Open macro architecture standard
- Shared GitHub repository
- Version-controlled macro framework

- Industry-wide automation patterns