AI Assistant Agent

Overview

The AI Script Assistant is an in-app coding partner built directly into the ARC script editor. Instead of switching to a browser, copying code from a chatbot, and renaming variables to match your project, you ask the assistant a question from inside ARC and get back a script that already uses the names you defined.
It generates JavaScript or Python (you pick), respects the ARC scripting API, and remembers the conversation so each follow-up builds on the last reply.
***Note:** This is not a chatbot, and it will not write memes, poems, or hold casual conversations. The ARC AI Script Assistant is purpose-built for robot programming and runs on an extensive GPU backend that searches significant project data storage on every request — your globals, EZB ports, robot parameters, controlCommand() targets, and the full ARC scripting API. Sending a "hello" still spins up that pipeline, wastes resources, and burns credits for nothing. Use it for what it's optimized for: writing, fixing, refactoring, and explaining ARC JavaScript and Python scripts.

How It Works

1

Ask in Plain Language

Open the assistant from any script editor and type your request — “sweep the servo from 1 to 180,” “fix this script, it hangs after the second loop,” or “explain what this code does.”
2

ARC Sends Project Context

Your prompt is bundled with live project state — globals, EZB ports, robot parameters, controlCommand() targets, and the current script — so the AI has ground truth instead of guesses.
3

Paste, Run, Iterate

The reply comes back in a fenced code block, ready to drop into your script. Don’t like it? Ask for changes — the conversation continues until the code is exactly what you want.

Why It’s Different

Generic AI chatbots invent function names, hallucinate APIs, and force you to find-and-replace placeholder variables before the code will run. The ARC AI Script Assistant ships with the full ARC scripting reference, plus your project’s authoritative state, on every turn. The code it returns uses real APIs and real names that already exist in your project.

What You Get

Project-aware code Scripts use the global variables, port names, and skill commands that already exist in your project.
JavaScript or Python The assistant targets only the language you picked — no mixed-dialect output, no surprise EZ-Script.
Fix, refactor, explain Paste broken code and ask for a fix. Paste working code and ask for a simpler version or an added feature.
Multi-turn conversations “Make it slower.” “Now add a sensor check.” Each follow-up builds on the previous answer.
Full conversation history Every prompt, reply, attached script, token count, and credit cost is saved and searchable in your account.
Transparent credit billing Pay only for tokens you actually use. Failed requests are free. Per-turn cost shown in your history.

Who Benefits

- **Beginners** — describe a behavior in English and get a working script without learning the full scripting API first.
- **Experienced builders** — skip the boilerplate. Generate a sweep, a sensor loop, or a state machine in seconds, then refine it.
- **Educators** — students learn by asking the assistant to *explain* what code does, not just by copying it.
- **Project owners** — refactor old scripts, port between JavaScript and Python, or document existing code on demand.

Ready to try it?

The AI Script Assistant ships with Synthiam ARC. Download ARC, open any script, and ask the assistant for the code you want.
Get ARC →
View AI Credits

How To Use Scripting Agent

Why this matters: The AI does not see your wiring, your servo limits, or what your skills are for. It sees the names and descriptions you write. A port called `d2` with no description is just `d2`. A port called `d2` described as *"left shoulder servo, range 30–150"* becomes a precise, hard constraint that the AI will respect.
Reverting Code: The AI will change any existing scripts based on your requests. If you ever wish to revert to a previous version of your code, view your AI Usage Log to see all changes and AI activity.

Project Configuration

1

Describe Your Global Variables

When you call `setVar()`, pass a third argument that explains what the variable means. The assistant reads this description as authoritative meaning. Without it, the AI is left guessing what `$mode` or `$batt` represents.
// Weak — AI has no idea what this is for setVar("$mode", "idle"); // Strong — AI now knows the purpose AND the valid values setVar("$mode", "idle", "Current robot behavior mode. One of: idle, patrol, follow, sleep."); setVar("$batt", 12.4, "Current battery voltage in volts, sampled every 5 seconds.");
2

Fill Out the AI Description on Every Script

Every script in ARC has an **AI Description** field in its configuration screen. Use it to write one or two plain sentences describing what the script does and when it runs. This is what the AI sees when it considers calling, modifying, or referencing that script.
× Weak: (empty) ✓ Strong: "Runs when the bumper sensor triggers. Stops all motors, backs up 30cm, then turns 90 degrees right."
3

Describe Auto Position Actions

Every Auto Position action has a description field. Fill it in. The AI uses these descriptions to choose the right action when a prompt asks for a pose or motion — without them, it can't tell *Wave* from *Salute* from *Point*.
× Weak: Action1 ✓ Strong: Wave — "Raises right arm to shoulder height and waves the hand left-right twice. Returns to home pose. Takes ~3 seconds."
4

Describe EZ-Bits in the Robot Designer

If your robot is built from EZ-Bits, open each bit in the Robot Designer and describe what its servos do, where they're located, and what ports they use. The EZ-Robot JD Humanoid template is a great reference — you'll see entries like *"D4 — left arm elbow"*, *"D5 — left arm wrist rotate."*

✓ Good descriptions look like: D0 — head pan (left/right), center=90, range 20–160 D1 — head tilt (up/down), center=90, range 60–120 D4 — left arm elbow, straight=90, fully bent=30 D12 — right gripper, open=70, closed=145

5

Use the Port Descriptions Tab (No EZ-Bits Needed)

Not using EZ-Bits? Open **Project → Properties → Port Descriptions**. Add each port you use and write a description of what it does. The AI reads this list the same way it reads EZ-Bit descriptions — it's the canonical "what is plugged into what" reference.

✓ Example port descriptions: D0 — pan servo, scanning lidar mount, range 0–180 D1 — tilt servo, scanning lidar mount, range 30–120 D8 — relay for headlight LEDs ADC0 — battery voltage divider, multiply by 3.2 for volts v0 — left drive motor, −100=full reverse, 100=full forward v1 — right drive motor, −100=full reverse, 100=full forward

6

Rename Robot Skills to Match What They Do

Robot skills are addressed by name through `controlCommand()`. Generic names ("Servo Movement Panel," "Soundboard 1") tell the AI nothing — descriptive names tell it everything. **Right-click any skill's title bar and choose *Rename***, then give it a name that describes its purpose.

× Generic: "Servo Movement Panel" → controlCommand("Servo Movement Panel", "Run") ✓ Descriptive: "HeadServo" → controlCommand("HeadServo", "Run") × Generic: "Soundboard 1" ✓ Descriptive: "GreetingSounds"

Prompting Tips

Project setup gets the AI the facts. Good prompting gets the AI the *intent*. The clearer your request, the closer the first reply lands to what you actually wanted.

1. Be specific about what the script should do

State the action, the inputs, the outputs, and any timing. "Move the arm" is ambiguous. "Sweep the left elbow from 30 to 150 degrees over 2 seconds, then back" is buildable.

× Weak: "make the robot wave" ✓ Strong: "Use the Auto Position 'Wave' action, then return to home pose. Speak 'hello' through the SayEZB skill while waving."

2. Name the skills, ports, or variables you want used

If you already know which skill or port should do the job, say so. The AI will use exactly what you named instead of guessing.

✓ Strong: "Read ADC0, convert to volts using the formula in the port description, and store it in $batt every 10 seconds."

3. State the trigger and the stop condition

Tell the AI when the script starts, when it stops, and what should happen in between. Loops without exit conditions are the #1 source of bad generated scripts.

× Weak: "patrol the room" ✓ Strong: "Drive forward at speed 40 until the front sonar reads under 30cm, then stop and turn right 90 degrees. Repeat until $mode is no longer 'patrol'."

4. Ask for a fix on the existing script — don't start from scratch

If you have working code that's *almost* right, leave it in the editor and ask for the specific change. The assistant gets the current script as context — it will modify what you have instead of rewriting it.

✓ Strong: "This script works but the head jitters at the end. Slow the last servo move down to take 1 second."

5. Ask for one thing at a time

Long compound requests confuse the AI and produce monolithic scripts that are hard to debug. Build behavior in small turns — get a working sweep, then ask to add a sensor

check, then ask to wrap it in a loop. Each turn produces tested, focused code.
6. Use follow-up turns to iterate
The assistant remembers the conversation. After the first reply, just say "make it slower," "add a check for the bumper," or "use Python instead." You don't need to restate the whole problem.
7. Ask the assistant to explain code you didn't write
Inherited a script from a community project? Paste it in and ask "explain what this script does, line by line." The assistant will walk through it using your project's real variable and port descriptions.
8. Pick the right model for the job
Use a small, fast model for quick fixes and one-line changes. Use a larger reasoning model when you need multi-step logic, careful constraint handling, or a refactor across a long script. You only pay for the tokens you use — matching model to task saves credits.

Quick Checklist

- Every global variable has a description in its `setVar()` call
- Every script's AI Description field is filled in
- Every Auto Position action has a description
- EZ-Bits or Port Descriptions list every port you use, with purpose and range
- Every robot skill is renamed to describe what it does
- You write specific prompts that name the inputs, outputs, and stop conditions
- You iterate one change at a time in the same conversation

Project Configured. Time to Build.

Open any script in ARC, click the AI Script Assistant button, and describe what you want. The work you just did makes every reply better.
Get ARC →
View AI Credits

How To Use Chat Agent

Talk to an AI model about your ARC project. Ask questions about robot skills, control commands, scripting, and project globals. The assistant pulls live context server-side so its answers stay grounded in your current project.
Heads up: Each request consumes credits from your Synthiam account. Your remaining balance is shown in the status bar after every reply.

Getting Started

You will find the robot programming AI Script Assistant in every JavaScript and Python script editor screen for any robot skill that supports scripts. This means you can open a script to edit, type in what you want the script to do, and the AI agent will create the script for you. As you will see in this manual, there are things you can do to inform the AI about your robot. Because the AI cannot see your robot, it needs to have predefined details, such as what servo ports, ranges, etc.. This information informs the AI about what servo and EZB port each joint is on, for example.

1. Pick a model
   Open the *Model* dropdown and choose the AI model you want to use. Models marked with a usage level, such as Pro, indicate their tier.
2. Type your prompt
   Enter your question in the input box. Multi-line prompts are supported, and you can press Enter to add a new line.
3. Send
   Click Send, or press Ctrl+Enter to submit.
4.

Read the reply
The response appears in the transcript with full markdown formatting, including fenced code blocks.

Keyboard Shortcuts

Shortcut	Action
Ctrl + Enter	Submit prompt
Enter	Insert a new line in the prompt without submitting

Toolbar Controls

Model

Dropdown of available AI models. The list is fetched from the server when the chat opens. Send is disabled until a model is selected.

Refresh

Re-pulls the model list and your credit balance. Use this when a new model has been added mid-session, or after purchasing credits in the browser.

Manage Credits

Opens the Synthiam AI credits page in your default web browser so you can review usage or purchase additional credits.

Status Bar

Displays loading state, credit cost of the most recent reply, your remaining balance, and any errors, which appear in red.

What the Assistant Knows

Every time you send a prompt, the chat automatically includes the following context with your request:

- Conversation history
  All prior turns in the current session, including any code the assistant has already suggested.
  Client
- Project global variables
  A snapshot of every global variable in your project, including name, value, type, description, and array entries.
  Client
- Project name
  The filename of the currently open project, or `Unnamed` if the project has not been saved.
  Client
- Control commands and robot skill info
  Pulled server-side through tool calls when the assistant needs them.
  Server

Synthiam documentation
Retrieved server-side through tool calls when relevant to your question.
Server

Tips for Good Results

Be specific

Mention the robot skill or control by name. “How do I make the Auto Position skill loop?” is better than “How do I loop?”

Reference your variables

The assistant can see all your project globals, so ask about them by name and it will use the current values.

Iterate

Conversation history is preserved. Refine an answer with follow-up prompts instead of starting over.

Use code blocks for code

If you are pasting an error or snippet, wrap it in triple backticks so the assistant parses it correctly.

Compared to the AI Script Agent

AI Assistant Chat keeps the full response, including code blocks, inline in the transcript. Use it for questions, explanations, and exploratory conversations.
AI Script Agent strips code from the chat and pushes it directly into a script editor. Use it when you want generated code dropped straight into a runnable script.

Troubleshooting

The Send button is disabled

No AI model is selected, or the model list failed to load. Click **Refresh** to try again. If the status bar shows “No AI models available,” check your internet connection and Synthiam account status.

I see a red error in the status bar

The request failed. The error message includes a public code you can reference. Common causes include an expired session, insufficient credits, or a temporary server outage. Click **Refresh** and try again.

My credit balance never appears

The balance fetch is best-effort and will silently fail if the server cannot be reached. It does not block you from sending messages. Click **Refresh** to try again, or check the credits page through **Manage Credits**.

The transcript is blank after I send

The transcript page may still be loading. Any prompts sent before it finishes are queued and flushed automatically once the page is ready. Give it a moment.

Project Configured. Time to Build.

Open any script in ARC, click the AI Script Assistant button, and describe what you want. The work you just did makes every reply better.
Get ARC → View AI Credits

Converting EZ-Script to JavaScript or Python

ARC’s AI Assistant Agent can help convert older EZ-Script code into JavaScript or Python directly inside the robot skill code editor.

The process is straightforward: copy the existing EZ-Script, paste it into the code editor for the language you want to use, and ask the AI Agent to convert it. This gives the AI the original script directly inside the editor so it can rewrite the code using the selected programming language.
Tip: After the AI converts the script, review the new code before running it. Some scripts may need small adjustments if they rely on older EZ-Script behavior, custom variables, robot skill names, or project-specific logic. You can continue to ask the AI Script Assistant to fix any issues that you experience after the conversion.

Why Convert EZ-Script?

EZ-Script was the original scripting language used by ARC. Modern ARC projects use JavaScript and Python because they are more powerful, more stable, faster, more effecient, more widely known, and better suited for advanced robot programming.
Converting your EZ-Script code helps modernize older robot projects while keeping the original behavior, logic, and automation that you already created.

JavaScript

A great choice for most ARC scripting because it is fast, flexible, and commonly used in modern applications.

Python

A popular choice for AI, automation, data processing, and readable robot programming logic.

AI Assisted

The AI Agent can read the code you pasted into the editor and rewrite it into the target language for you.

How It Works

You will copy the EZ-Script from the original robot skill script editor and paste it into the JavaScript or Python editor. Then, use the AI Agent tab on the right side of the editor to ask for the conversion.
Basic Conversion Process

1. Open the robot skill that contains the EZ-Script code.
2. Open the code editor for that robot skill.
3. Select all of the EZ-Script code.
4. Copy the EZ-Script code.
5. Switch to the JavaScript or Python editor.
6. Paste the EZ-Script code into the new language editor.
7. Ask the AI Agent to convert the code.
8. Review the converted code before running it.

Convert EZ-Script to JavaScript

Use this process when you want to convert an EZ-Script into JavaScript.
Steps

1. Load the robot skill's code editor that contains the EZ-Script.
2. Select all of the EZ-Script by pressing Ctrl + A, or right-click in the editor and choose **Select All**.
3. Copy the selected code by pressing Ctrl + C, or right-click and choose **Copy**.
4. Switch to the **JavaScript** editor window.
5. Paste the EZ-Script code into the JavaScript editor by pressing Ctrl + V, or right-click and choose **Paste**.
6. Open the **AI Agent** tab on the right side of the code editor.
7. Enter an instruction such as the example below.
8. Press the **Send** button.
9. Review the JavaScript code generated by the AI Agent.

Example instruction:
`Convert this EZ-Script code into JavaScript.`

Better JavaScript Conversion Prompt

For better results, give the AI Agent a little more direction:

```
Convert this EZ-Script code into JavaScript for Synthiam ARC. Keep the same robot behavior,
preserve the variable names where possible, and add comments explaining the converted logic.
```

Convert EZ-Script to Python

Use this process when you want to convert an EZ-Script into Python.
Steps

1. Load the robot skill's code editor that contains the EZ-Script.
2. Select all of the EZ-Script by pressing Ctrl + A, or right-click in the editor and choose **Select All**.
3. Copy the selected code by pressing Ctrl + C, or right-click and choose **Copy**.
4. Switch to the **Python** editor window.
5. Paste the EZ-Script code into the Python editor by pressing Ctrl + V, or right-click and choose **Paste**.
6. Open the **AI Agent** tab on the right side of the code editor.
7. Enter an instruction such as the example below.
8. Press the **Send** button.
9. Review the Python code generated by the AI Agent.

Example instruction:
`Convert this EZ-Script code into Python.`

Better Python Conversion Prompt

For better results, ask the AI Agent to keep the original robot behavior:

```
Convert this EZ-Script code into Python for Synthiam ARC. Keep the same robot behavior,
preserve the variable names where possible, and add comments explaining the converted logic.
```

Example Workflow

Original EZ-Script
Copy the EZ-Script from the original script editor.

```
Say("Hello, I am ready")
Servo(D0, 90)
Sleep(1000)
Servo(D0, 20)
Say("Done")
```

AI Agent Instruction
Paste the EZ-Script into the JavaScript or Python editor, then ask the AI Agent to convert it.

```
Convert this EZ-Script code into JavaScript for Synthiam ARC. Keep the same behavior and add
comments.
```

Important: The AI Agent may update the code in the editor. Always review the converted script before running it on a real robot, especially if the robot moves servos, motors, wheels, arms, or other physical hardware.

Tips for Better Conversions

Be Specific

Tell the AI Agent which language you want and whether you want comments added to the converted code.

Keep Skill Names Clear

If the script uses robot skills, make sure the robot skill names are still correct in the converted code.

Test Carefully

Run the converted script slowly and safely, especially when it controls movement, servos, or external hardware.

Useful prompt examples

- `Convert this EZ-Script into JavaScript and explain any changes.`
- `Convert this EZ-Script into Python and keep the same variable names.`
- `Convert this EZ-Script into JavaScript and add comments for each section.`
- `Convert this EZ-Script into Python and tell me if anything needs to be manually checked.`

After the Conversion

Once the AI Agent has converted the code, read through the new script and confirm that the logic still matches the original EZ-Script. Pay special attention to servo ports, robot skill names, global variables, delays, loops, and any commands that interact with hardware.
Recommended Final Prompt
After the first conversion, you can ask the AI Agent to check its work:

```
Review this converted script and compare it to the original EZ-Script behavior. Tell me if anything looks incorrect or needs to be manually verified before running it on my robot.
```