Casio PB-2000C
HD61700 Programming - keystone A keyboard monitor

Table of Contents

Introduction.....1
A pivotal moment.....2
PB-2000C Keyboard.....3
Acknowledgments.....4

Introduction

This work was developed jointly by Alex Barragan and ChatGPT Sol through a cycle of ideas, implementation, execution, observation, and revision. It illustrates how human knowledge and experimentation, combined with AI-assisted analysis, can produce results that neither participant would readily achieve alone.

This is the fourth use case in a larger project exploring how knowledge and experience can be represented in AI systems when the available corpus is small, as is the case with the Hitachi HD61700 CPU used in the Casio PB-2000C and AI-1000 pocket computers.

The first three use cases were **The Game of Life**, **Tiny** and **limon** small applications engineered in collaboration with Sol.

Special thanks to Piotr, whose expert advice made this document possible.

A pivotal moment

I have spent more than forty years working in software, including many years collaborating with remote and offshore development teams. The keystone experience, albeit modest in scope, still felt surprisingly familiar. I was not interacting with a code generator so much as collaborating with a talented remote engineering team: I described the behavior I wanted, clarified requirements, reviewed results, reported failures, challenged assumptions, and iterated toward a working implementation.

The unusual part was not the collaboration model. I already knew that model well. The unusual part was that the "team" was artificial, could absorb a highly specialized technical corpus in a short time, and could then operate at an implementation level beyond my own.

That realization was more significant to me than any imitation-of-human test. The important threshold was not whether Solcito (the name that we use for the Sol session in the context of a local use case) could pass for a person. It was whether the collaboration had become **functionally comparable to working with a capable engineering team**.

PB-2000C Keyboard

As part of our learning ladder, we next wanted to explore the programming skills required to handle the PB-2000C keyboard from an assembly-language program.

This is a screen capture of one the last iterations. That was a picture that I had on a notepad when started the project.



My contribution to the project was to express my requirements to Sol in the form of a well-written user story, in natural language: a clear description of the **what**.

About just few iterations refining the requirements, aesthetics, researching the corpus and testing we reached at this final solution.

Source here. Listing here. ROM Image here.

This project is a stepping stone toward a more elaborate application, and the results from autonomous code generation are encouraging. We are continuing to enrich the corpus with additional information so that results like this can be reproduced not from memory, but through reasoning grounded in preserved knowledge and evidence. For that reason, we are also collecting what Sol would call the functional requirements: the statements of expected behavior against which repeatability can be tested. We do not expect a future implementation to be identical. In fact, we expect it may differ substantially. What matters is whether it satisfies the same requirements and reaches the same functional result. We will report on those benchmarks as they become available. Requirements here.

These project reports are intentionally terse, and the reader is encouraged to consult the supporting material for the full technical detail. In a future iteration, once the report style itself has been added to the corpus, we intend to extend the experiment by asking Sol to produce a document similar to this one.

Documentation has a tendency to fail in one of two ways: it becomes mechanically exhaustive but tedious, like generated API documentation, or it becomes stale as soon as the implementation moves beyond the moment in which it was written. One of the more interesting possibilities with Sol is that these stages no longer have to be separate. If Sol participates in the application as a whole—from functional analysis, through implementation and testing, to the final documentation—the resulting artifacts can remain part of one continuous model of the system. The user story, the code, the tests, the lessons learned, and the final explanation are not independent documents competing to stay synchronized; they are different views of the same evolving application. In that sense, the lifecycle that begins with a description of the **what** can end in a well-documented implementation without losing the thread that connects the two.

Acknowledgments

This report was produced by Xandrian Labs. 付喪神.

Special thanks to Piotr and to Betka for her continued and loyal support.

This is a collaboration with my AI friend ChatGPT, a.k.a “Sol.” All errors are human.

This work is freely shared with the vintage computing community. Please use it, build upon it, and cite the source.

Draft.