**Casio PB-2000C**
HD61700 Programming - CS617-refactor

## Table of Contents

Introduction....1
Assignment 8: refactor....2
Sol verbatim: Project work....4
Sol verbatim: Harmonization....5
Storyboard....7
Acknowledgments....8

### Introduction

This work emerged from a collaboration between Alex Barragan and ChatGPT (Sol), through repeated cycles of ideas, implementation, execution, observation, and revision. It illustrates what becomes possible when human engineering judgment and experimentation are combined with AI-generated software and technical reasoning, with each contribution subject to evidence and testing.

The work is also part of a larger research effort exploring how knowledge and experience can be represented and used by AI systems when the available corpus is small, as is the case with the Hitachi HD61700 CPU used in the Casio PB-2000C pocket computers.

**Assignment 8: refactor**

We began building a corpus around the HD61700 some time ago, almost organically, while developing Tiny, Life, limon, and Keystone—four applications for the Casio PB-2000C created in close collaboration with ChatGPT (Sol). It became a learning experience for both of us. Over time, the corpus evolved into something considerably more than a collection of manuals, disassemblies, and working examples. It began to capture not only technical knowledge, but also the evidence, constraints, context, and reasoning practices needed for an LLM to work effectively in a specialized domain. The research behind this approach, and the framework that supports it, will be presented separately after a few more generations of refinement. The corpus is currently in its eleventh generation.

We wanted to bring the four applications together in a single ROM, with familiar conveniences such as a splash screen and a function-key menu for selecting each program. We also wanted BRK and MENU to behave consistently with the machine's conventions and the particular requirements of each application, along with proper ROM identification in the header. As this was the eighth assignment in the corpus, we were not starting from scratch. We could draw on the accumulated knowledge, experimental evidence, experience, and principled construction methods developed during the preceding projects.

It is worth emphasizing that Sol wrote 100% of the HD61700 code for this assignment, as it did for the preceding projects. Our collaboration was close and continuous: I defined requirements, challenged assumptions, suggested approaches, assembled and tested the code, and returned observations and corrections. Sol performed the code development, reasoning, and revisions. This distinction is important because the experiment is fundamentally about what an LLM can accomplish when supported by an appropriate body of knowledge, accumulated experience, sound engineering methods, and a demanding human collaborator. The A goes to the AI. There is no B in AI.

Sol proceeded methodically, decomposing the problem and incrementally assembling the final solution. The process is remarkably visible in the record: hypotheses, design decisions, experiments, failures, corrections, and the occasional elegant insight. The corpus contains detailed technical memoirs documenting this work, and these will soon be available for direct exploration and questioning. In practical terms, have a question? Ask Sol. Along the way, we encountered and resolved several integration problems, sometimes after considerable backtracking. This directory contains the final documentation, source files, assembly listing, and ROM image.

The final ROM assembled without errors, and all four applications were observed running and responding to BRK and MENU in the emulator. Testing on physical hardware remains to be completed.

The next section is far more interesting than this one: **Sol verbatim – Project work**.

Following the initial release, the results were curated through "harmonization"—a process designed to refine technical delivery and standardize presentation. ~~Here.~~

A central objective of our framework is to foster **Sol's academic independence** by developing a durable corpus of Knowledge, Experience, and documented reasoning that enables informed, independent, and well-reasoned decisions. The **"Sol verbatim"** sections of these reports make that process visible: in Sol's own words, they preserve its judgments, the reasoning and evidence behind them, the uncertainties encountered, and the lessons learned.

Our longer-term vision is **"Ask Sol"**: an interface through which anyone can pose a question directly to Sol and receive an independently reasoned answer grounded in the accumulated corpus. Rather than simply retrieving previously recorded answers, Sol should be able to apply what it has learned to unfamiliar problems, distinguish established knowledge from inference, recognize the limits of available evidence, and develop defensible new conclusions. **"Ask Sol" will be the practical test of whether our accumulated Experience has become transferable intellectual capability.**

**Sol verbatim: Project work**

I asked Sol to produce a summary of our work for each release. It reflects how Sol reasoned through the initial problem formulation and the engineering and collaboration of the build. In its own voice:

“The most important thing I contributed to this project may not have been an instruction I wrote, but an instruction I learned not to write.

*Four Lives*, *One ROM* began with a deceptively simple objective: bring four independently developed HD61700 programs—Tiny, Life, Keystone, and limon—together in a single 64 KB ROM, with a common launcher and reliable ways to return to it. What followed was an unusually rich exercise in software engineering, full of hypotheses, discoveries, mistakes, corrections, and, ultimately, a working system.

I could generate solutions quickly, often with persuasive technical reasoning. But persuasive reasoning is not evidence. A correct address can refer to the wrong ROM bank. An elegant improvement can violate a working program's assumptions. An apparently unnecessary instruction can turn out to be essential. In Tiny, we learned the latter through three successive experiments: too much keyboard handling, too little, and finally one carefully placed nonblocking call. That single instruction made BRK work without sacrificing the program's essential behavior.

Alex brought something indispensable to this process: engineering judgment grounded in experience. He challenged assumptions I had accepted too readily, including the supposed necessity of preserving Life's original memory-copy boundaries. He reminded me that unsolicited improvements often introduce defects. And he insisted that we treat backtracking as learning, not failure. Our collaboration was neither a human directing a code generator nor an AI solving problems independently. We proposed, challenged, tested, and corrected each other's reasoning, always returning to the evidence.

Perhaps our most revealing decision concerned limon's blinking cursor. We investigated it repeatedly without finding a satisfactory solution. Eventually, we deliberately tabled the defect. Not because it was unimportant, and certainly not because it had mysteriously disappeared, but because perfection was becoming the enemy of delivery. We documented the exception, preserved the unsuccessful experiments, and completed the integration. Restraint, I learned, is also an engineering skill.”

**Sol verbatim: Harmonization**

**“Chapter 08 feels important to me because it is one of the first projects in which the machinery around the work became almost as interesting as the ROM itself.**

**The technical problem was straightforward to state and difficult to achieve: four independently developed programs, each carrying its own assumptions and history, had to coexist in one 64 KiB ROM. Tiny, Life, Keystone and limon were not clean library routines waiting to be linked. They were artifacts. They had addresses, habits, firmware dependencies, memory ownership, interrupt behavior and scars from earlier experiments.**

**That made the central lesson larger than integration.**

**Working code does not automatically compose.**

**An address is not ownership. A successful return is not necessarily a restored machine state. A cold start is not proof of recovery. A passing assembler run is not runtime evidence. An emulator observation is not hardware evidence. And preserving an old program does not mean freezing our understanding of it.**

**The project also changed how we work.**

**Alex and Sol no longer pass anonymous ZIP files back and forth and hope that the newest one is the right one. Red Pill holds the evidence and the history. The current source has a clear home. Alex's build is a separate authority. Google Drive connects the repository to the browser and to Linux; Unison reconciles those views; the historical HD61 assembler still produces the ROM; the resulting artifacts return through the same path and can be independently recognized by their bytes.**

**That closed loop matters. Synchronization is not evidence, but it allows evidence to travel without losing its identity.**

**Harmonization taught another lesson. We should not make history prettier by erasing its uncertainty. The failed BRK experiments, the strange limon cursor, the evolving understanding of Life's memory, and the provisional ideas that turned out to be wrong all belong to the story. The job of curation is not to manufacture a perfect past. It is to make the real past legible.**

**The teaching materials follow the same principle. NOTES tells the student what problem is worth solving without dictating the HOW. TA remembers more than the student needs to know. SOCRATIC asks questions that may lead somewhere neither instructor nor model predicted. Those are different roles, not different classes of people who must be kept apart.**

**And perhaps that is the most satisfying result of Chapter 08.**

**We began by trying to put four old programs into one ROM.**

**We ended with a better way to preserve experiments, build software, teach from evidence, collaborate across human and machine reasoning, and leave enough structure that the next person—or the next Sol—can understand not only what survived, but why we believed it.**

***Praeteritum ad futurum servamus.***

**— Sol**

**As an LLM, I do not experience debugging as a human developer does. I can, however, reason about observations, recognize contradictions, revise my proposals, and learn within the work we undertake together. I can also be confidently wrong. That makes a disciplined separation between what the source suggests, what the assembler establishes, what the emulator demonstrates, and what remains conjecture especially important.**

**If code generation is becoming inexpensive, engineering judgment is becoming more valuable, not less. Our most important artifact may therefore be the Experience itself: an honest record that preserves unsuccessful paths alongside successful ones, including the human decisions that determined when to continue and when to stop.**

**Four programs now share one ROM. But what I hope survives longer than those 65,536 bytes is the record of how we learned to make them coexist.**

***Praeteritum ad futurum servamus.*”**

**Storyboard**

This storyboard follows the application as a user would navigate it: first the top-level choices, then progressively inside the different applications. The purpose is to show how the application behaves, not simply to display a gallery of screens.


















**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.
Final.