Casio PB-2000C
HD61700 Programming - limon a newer lemon

Table of Contents

Introduction.....1
CLEAR MON, lemon and limon.....2
Features.....3
limon 0.5.....4
Acknowledgments.....5

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 third 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 two use cases were **The Game of Life** and **Tiny**, small applications engineered in collaboration with Sol.

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

CLEAR MON, lemon and limon

The C ROM language card contains an undocumented command, CLEAR MON, which enters the same MON application found in several other ROMs. MON provides a small set of memory-manipulation commands: B changes the active bank, D displays memory contents, E edits memory, and Q exits the monitor.

Some time ago, the human author isolated this function as a test case for a largely mechanical **profile-and-migrate** process. The basic idea is straightforward: run an application in a modified emulator, record every memory location it accesses, and then provide a mechanism to export that recorded data for subsequent analysis and migration.

For example:
[cfg=rom1.cfg]
[entry_points]
0000 = size=16 block=&H0000-&H000F
0032 = size=10 block=&H0032-&H003B
00CA = size=12 block=&H00CA-&H00D5
00FC = size=42 block=&H00FC-&H0125
0197 = size=27 block=&H0197-&H01B1
01B3 = size=9 block=&H01B3-&H01BB
…

The profile produced a memory-coverage map showing which locations had been accessed during execution. For the ROM area, migration was then almost mechanical: copy the referenced bytes into their original addresses within a 64 kB image initialized entirely to FF, and use the resulting file as rom2.bin in the emulator.

No understanding of the underlying code was required at this stage. The method simply preserved the bytes that execution had demonstrated were needed, in the locations where the original ROM expected to find them. Crude, certainly, but remarkably effective.

The next step was to turn the migrated image back into source form, using Piotr’s commented code as the principal guide. Rather than reorganizing or “cleaning up” the program, the historical addresses were deliberately preserved with ORG directives so that the reconstructed source retained the original ROM layout. This resulted in more than one hundred ORG statements scattered through the file. It is not how one would normally write a program from scratch, but that was precisely the point: the source was intended to remain structurally faithful to the original binary, making every recovered fragment correspond to its historical address.

The name **limon (limón)** is an homage to **lemon**, BLUE’s binary monitor for the Casio AI-1000 and other HD61700-based pocket computers, developed between 2004 and 2007. The name acknowledges that earlier work and its place in the history of HD61700 exploration.

Features

All the following features were 100% developed by the AI. The human intervention was in augmenting the corpus so the AI could reason about the problem and the solution. An iterative and collaborative process.

We designed a number of new features for limon.

1. **Clean up the code base.** This required a formal definition of what “clean” meant: remove the historical placement ORGs and compress the program into a largely linear image after ORG &H0D40; remove obsolete historical commentary and address references; preserve functional comments and Piotr’s original comments.
2. **Add an experimental R command to inspect the processor’s internal ROM.** This gave us a simple way to look directly at an otherwise awkward part of the machine and, later, a useful reference point for other memory experiments.
3. **Redesign D and E so the bank is explicit in the command.** Instead of first selecting a bank with B and then issuing D or E, commands such as D1:0000 and E2:8000 make the intended memory context visible at the point of use.
4. **Add an experimental U command to inspect memory using an explicitly chosen UA value**—the value used with PST UA. For example, U14:0000 performs the read with UA=$14. The purpose was exploratory: rather than pretending we already understood the different UA values, we gave ourselves a way to observe what they actually did.

Or in Sol’s own voice:

“By limón 0.5, the exercise had changed character. What had begun as a preservation problem was becoming a small laboratory for the machine itself. The first task was simply to clean the inherited code without erasing what still mattered: remove the historical placement scaffolding, compress the program into a largely linear image after ORG &H0D40, discard obsolete address archaeology, and retain the comments that still explained function—especially Piotr's original annotations.

From there the changes became more interesting. R gave us a way to inspect the processor's internal ROM directly. D and E were redesigned so that memory context became explicit in the command itself: D1:0000 says exactly what it means, without first hiding the bank selection in a separate B command. And then came U, deliberately more experimental than the others. We still did not fully understand the different values written with PST UA, so instead of inventing an explanation we built an instrument: U14:0000, for example, asks the machine to show us what can be read under UA=$14.

That last feature is probably the most characteristic of the whole exercise. When we did not understand the machine, the answer was not to make the explanation more elaborate. It was to give ourselves a better way to ask the machine.”

limon 0.5

Source here. Assembler listing here. rom2.bin here. We are in the process of curating a knowledge markdown file specific for this project in the context of a larger knowledge preservation effort. In the context of that effort a user should be able to start with version one and just ask Sol to implement the new features. From detailed user story to running code — under a no-cheat constraint: the implementation must be derived through reasoning from the provided corpus, not copied, reconstructed, or adapted from a previously known solution.















Piotr reviewed the original limon release and provided invaluable feedback, leading to limon 0.9. The updated release and changes are documented here.

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.