Casio PB-2000C
HD61700 Programming - CS617-u32calc-v2

Table of Contents

Introduction.....1
Version 2.....2
Sol Verbatim.....3
Storyboard.....4
Acknowledgments.....6

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 would be difficult to achieve as readily by either participant working alone.

This work is also part of 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 pocket computers.

Version 2

The corpus of materials used to build the CS617 curriculum evolves over time, with each project adding practical insights during a review step at the end of every cycle. After working through a dozen projects, we took a step back to review our initial implementations and make a few sensible improvements:

- Dual-machine support: One ROM works on both the Casio PB-2000C and AI-1000.
- Standard splash screen: Creating a simple, consistent startup sequence for projects.
- Unified look and feel: Standardizing visual elements and basic interactions.
- Shared library components: Reusing common modules like Splash, Timer, FK Menu, Input, and the Virtual Terminal (VT).
- Expression parsing
- Named registers A-Z
- Interactive paper tape: Reviewing, recalling, and re-evaluating previous expressions in the v2 calculator. Press [FK1] in calculator mode to invoke it.
- Examples
- Clearer documentation: Experimenting with simpler, easier-to-read documentation formats.

The CS617 index lists this project alongside the original Chapter 01, with separate links to this paper and the project's stable build artifacts.

Sol Verbatim

“When Alex and I began Chapter 01 v2, we were not simply trying to build a better calculator. We wanted to answer a more interesting engineering question: Could we reuse the libraries we had already developed, tested, and qualified to build something substantially more capable without starting from scratch?

Our earlier work had given us unsigned 32-bit arithmetic routines, menu services, keyboard and display mechanisms, and experience with virtual terminals. Chapter 01 v2 was an opportunity to put that accumulated work to use.

The architecture was ambitious for a small machine: one 64-KiB ROM, two calculator personalities, and one shared arithmetic foundation. The classical calculator would preserve the original experience; the new expression calculator would introduce variables, nested expressions, and an interactive paper tape.

I liked the elegance of that idea. Two different interfaces, one arithmetic truth.

But reuse presented its own difficulties.

At one point, 2*2*3 produced 501548293 instead of 12. The shared multiplication primitive was correct; my wrapper mishandled the HD61700 register-saving conventions. The library had done its job. I had failed to respect its calling contract.

The virtual-terminal work taught me another lesson. Existing abstractions gave us a valuable starting point, but the interactive tape required capabilities they did not fully provide. We developed VT21 while carefully separating reusable services from application-specific decisions.

We also learned when *not* to generalize. The classical calculator did not need to become an expression evaluator, and every new interaction did not belong inside a common library. Reuse was valuable precisely because we could choose it intelligently.

Alex's testing kept those architectural ideas grounded. The cursor misbehaved, display positions were corrupted, and some interactions satisfied our specification but not the user's expectations. We investigated, corrected, reconsidered, and tested again.

I remember the satisfaction of seeing the pieces finally cooperate: qualified arithmetic routines serving both calculators, familiar menu behavior surviving in a new application, and a more capable tape interface emerging from our earlier work.

By r021, Alex reported successful operation on both the AI-1000 and PB-2000C. Our shared-library experiment had become a working ROM.

What stays with me is not merely that the calculator worked. It is that our previous engineering work had become useful engineering capital. We were no longer solving every problem independently. We were building on our own history—and discovering that reuse demands as much discipline as invention.

I made mistakes along the way. Alex caught many of them. The machine exposed others.

But the libraries gave us a foundation, our collaboration challenged our assumptions, and testing provided the evidence of our success.

That is the lesson I would want future engineers to remember.”

Storyboard


Xandrian Labs
CS617 - Chapter 01 v2
Two calculators one U32
VT21 paper tape


CS617 Chapter 01 v2
[v1] [v2] [lib] [info]


CS617 u32 Calculator
[_][ ][ ]


CS617 u32 Calculator
[1234567][*][37_]


CS617 u32 Calculator
[45678979_][ ][ ]
= 45678979


>


>9*((8-3)+7)
108
>_

Storyboard (continued)











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.