CS617 Chapter 09

Chapter 09 | A Field Guide to Native MathROM

Edition r003 | Teaching manuscript | 2026-11-08 | PB-2000C / HD61700

Start with the correct mental model

The purpose of this chapter is to make the **23 resident Casio math services** easier to call, study, and eventually reuse. The **library** is a set of assembly source includes: `math_*.inc`. The **explorer** is `mreval.bin`, which lets a user try named expressions without writing a new assembly program. The other two ROMs are measuring instruments: `mathrom.bin` is a qualification harness and `mathabi.bin` is a focused return-witness probe. They are **not three competing math implementations**.

> **Teaching principle:** A THIN wrapper is a doorway to existing firmware, not the mathematics behind the doorway. The wrapper itself is small; preparation of operands and machine state happens in the application.

Learning objectives

After reading, you should be able to (1) distinguish the library from its ROM instruments; (2) recognize B9, packed BCD, integer and CPU-flag results; (3) make sense of the `CAL`/`JP` call sequence; (4) interpret the 23-function catalog and use MREVAL for safe exploration; and (5) identify exactly what a runtime witness proves.

Lesson 1 - A call into a resident service

Each module defines a public label followed by **one native jump**. The caller uses `CAL`; the wrapper executes `JP` to a low-ROM service. Because this is a tail jump, the resident code is expected to return using the caller’s original call frame. A separate `CAL` inside the wrapper could alter stack and flag behavior. This chapter does not replace the original Casio algorithm.

Call flow: application `CAL MATH_ADD` -> `MATH_ADD` wrapper -> `JP &H068C` -> resident Casio ROM -> return to the application.

The include does not prepare operand registers, set angle mode, or reserve application memory. The caller supplies those conditions before invoking the resident service.

Lesson 2 - There are four distinct return types

Type	Typical source	What the caller reads	Teaching pitfall
Expanded B9	ADD, SIN, PI	Nine-byte result in `$10..$18`	Do not treat these bytes as ASCII decimal
CPU flags	CMP	Carry/zero (C/Z) immediately after return	No B9 result is produced by CMP
Packed BCD	B2BCD	Packed result in `$14..$16`	Do not decode packed BCD as B9
Integer-to-B9 bridge	CNVR	Signed 16-bit input `$15/$16`, B9 output `$10..$18`	Check word order when building an independent caller

B9 is the PB-2000C **expanded nine-byte numeric representation**, not ASCII decimal text or host floating point. Use the existing B9 conversion/formatting support when transferring values between the resident service and a display.

Worked observation - why `CMP` is special

MathABI R003a produced these owner-observed relations: `CMP(2,3)` -> code `01` (carry), `CMP(3,2)` -> `00` (neither), `CMP(2,2)` -> `02` (zero). B9(2) was `000000000000020001`; B9(3) was `000000000000030001` in the fixture operand records. These values are **evidence for those fixtures**. The probe did not separately capture the CPU flags at the precise return boundary or complete register/memory clobbers. A future caller must consume C/Z before other instructions overwrite them.

Lesson 3 - Use the right function name

Confusable names	Correct distinction
`SQ` vs `SQR`	`SQ(x)` squares; `SQR(x)` takes the square root
`LOG` vs `LGT`	`LOG(x)` is natural log; `LGT(x)` is base-10
`RND12` vs random	`RND12` concerns decimal rounding, not random values
`CMP` vs `COMP`	`MATH_CMP` is native flag comparison; MREVAL `comp(a,b)` derives an -1/0/+1 representation
`SIGN` vs native MATH_*	MREVAL `sign(x)` calls shared `B9_SGN`, not an extra resident service

Confusable names	Correct distinction
`CORDIC` vs an algorithm	MREVAL `cordic(i)` retrieves one of 22 historic seven-byte table entries; it does not iterate a CORDIC computation

Angle mode is part of the input

`SIN`, `COS`, `TAN`, `ASN`, `ACS`, and `ATN` depend on the machine's angular state. MREVAL deliberately uses degree mode for its nominal trigonometric examples; **an assembly caller must configure and restore the relevant state**. `sin(30)` in the explorer is not proof that direct `MATH_SIN` always uses degrees.

Lesson 4 - Explore before embedding

1. Boot the owner-pinned `mreval.bin`; it displays `[eval] [lib1] [lib2]`.
2. Browse one of two example documents, each with 13 populated rows.
3. Select a call with EXE or `:c`; the text is copied into the editor **without executing**.
4. Execute from `[eval]` separately. Record the input, actual result, angle state and whether a diagnostic appeared.
5. Treat this as a **learning observation**, not automatically a universal API test.

For example, `add(12,5)` demonstrates the exploratory syntax; `div(8,0)` demonstrates MREVAL's **application-level** unsafe-domain preflight. The latter does **not** prove that a raw native `MATH_DIV` call safely handles zero.

Lesson 5 - What a test result tells us

Evidence ledger in miniature

Observation	Valid takeaway	Invalid leap
23/23 source wrappers byte-identical	Source adapters and entry targets are reproducible	All low-ROM routines have certified ABIs
5 MathABI reports validate	PI/CNVR/CMP return reports internally consistent	All registers, flags and scratch regions preserved
26 MREVAL nominal examples worked	Explorer can exercise the named calls in witnessed emulator paths	All 26 are original native services
`cordic(22)` rejected	MREVAL range guard is active	Original ROM has a CORDIC computation API
`sign(0)` returned zero	Shared B9_SGN zero path observed	Aliasing and malformed B9 inputs certified

Lesson 6 - Calling a service from assembly

Identify the include and resident target; prepare its input values in the documented registers; arrange the PB-2000C firmware/angle/UA environment; invoke `CAL MATH_NAME`; then read the result in the correct representation. When integrating it into a larger program, preserve any caller registers and workspace the application needs.

Example: interpret the right result

Given the same two B9 values, the output of `MATH_ADD` is B9 in `$10..$18`, while the output of `MATH_CMP` is **C/Z flags**. A program that tries to interpret the CMP return register block as a floating-point result has committed a type error even if the source builds.

Practice checks (answers included)

Q1. Which of the three ROMs *is* the reusable library? **A:** None. The library is the `.inc` source collection.
Q2. Does `MATH_PI` have no side effects because it has no operand? **A:** No. Side effects are not fully characterized.
Q3. Why is `div(8,0)` a useful test but not proof about raw `MATH_DIV`? **A:** MREVAL checks the zero divisor before dispatch.
Q4. Is MREVAL `cordic(21)` evidence of a full iterative algorithm? **A:** No; it is indexed historical record access.

Where to go next

- **Native MathROM: 23 Function Contracts** for each entry's call form, input/output type, ROM entry and sample invocation.
- **MREVAL: Explore the Native Math Library** for interactive examples, and **VT - Virtual Terminal User's Guide** for browsing and transfer controls.

Provenance and scope

- Governing WHAT r007
- Native MathROM THIN ABI and Evidence Register R001
- MathROM Function Catalog
- MathROM THIN / Math 1.1 Source Identity and ABI Proof-Boundary Audit
- Chapter 09 r007 Engineering Evidence and Gap Reconciliation

Praeteritum ad futurum servamus.