

Chapter 62 - Assets, ROM Data, And Build Hygiene

This chapter is deliberately careful. A porting tree may have a legal and repeatable way to derive assets outside the guide, but this book does not teach extraction and does not require the reader to own or process game data.

The useful IE lesson is asset discipline.

62.1 Name Assets, Do Not Scatter Them

The game core asks for assets by stable names. The IE layer decides whether those names are found in a packed image or through the File I/O device.

That gives the port three useful properties:

- The game code does not know the storage medium.
- The pack table can be checked for size and overlap.
- A missing or oversized asset fails explicitly.

62.2 Keep Ranges Apart

The packed image uses separate regions for the loader, table of contents, asset data, game image, and texture store. The tests reject layouts that overlap the texture window or exceed the allowed pack data range.

The rule is simple: every large region needs a named home before the programme grows.

62.3 Byte Order

The M68K side is big-endian, and the pack header follows that rule. Texture and audio devices may have their own device formats. The port must convert data at the boundary, not guess later.

Use one byte-order rule per file or device contract, write it down, and test it.

62.4 What The Reader Does Not Need

The reader does not need:

- ROM file names.
- Extraction commands.
- Commercial asset layouts.
- Generated binary dumps.
- External build scripts.

The reader needs the IE pattern: named assets, deterministic layout, range checks, byte-order rules, and explicit failure.

62.5 The General IE Lesson

Treat assets as part of the machine contract. A large IE programme should know what data exists, where it can live, how large it may be, and how failure is reported. Do not let asset handling become an unwritten side channel.