Aves Documentation Plan

Table of Contents

1. Introduction

This document defines the structure, style, and organization for the Aves project documentation. It covers both hardware and software manuals, establishing consistent standards across all documentation.

NOTE For directory structure and organization of front matter components, refer to the Organization Plan in aves-org-plan.adoc.

2. Documentation Standards

2.1. Document Organization

2.1.1. Section Hierarchy

Level	Marker	Usage
Level 1	=	Main topics, chapters
Level 2	==	Major sections
Level 3	===	Subsections
Level 4	====	Detailed points

2.1.2. Section Components

Each section must contain:

1. Opening Narrative
    - Provide context and overview

    - Explain key concepts
    - Define terminology
    - State purpose and scope
2. Tables Section (if applicable)
    - Present structured data
    - Use consistent formatting
    - Include clear headers
    - Group related information
3. Examples Section (if applicable)
    - Provide practical implementations
    - Include relevant code snippets
    - Add explanatory comments
    - Show multiple cases where needed

3. Style Guide

3.1. Writing Style

3.1.1. Language and Grammar

Table 1. General Principles

Principle	**Guidelines**
Language	Use British English spelling and conventions Use clear, direct language Use active voice where possible Avoid jargon unless technically necessary
Paragraphs	One main idea per paragraph Use transition sentences between paragraphs Keep paragraphs focused and concise
Technical Terms	Define on first use Maintain consistent terminology Include in glossary Use approved abbreviations
Gender References	Use gender-neutral language Avoid gender-specific pronouns Use plural forms where possible

3.1.2. Technical Writing

Table 2. Technical Content Guidelines

Element	Requirements
Procedures	Number steps sequentially One action per step Include prerequisites Note expected outcomes
Concepts	Define before use Progress from simple to complex Include relevant examples Cross-reference related topics
Warnings/Notes	Place before affected content Clear, concise wording Highlight critical information Use appropriate admonition type

3.2. Naming Conventions

3.2.1. Hardware Manual Conventions

Table 3. Hardware Signal and Register Naming

Type	Convention	Examples
Register Names	ALL_CAPS	`CONTROL_REG`, `STATUS_REG`
Bit Fields	ALL_CAPS with bit position	`MODE[2:0]`, `STATUS[7]`
Signal Names	CamelCase with _N for active-low	`Clock`, `Reset_N`, `ChipSelect_N`
Bus Signals	CamelCase with width	`DataBus[7:0]`, `AddressBus[15:0]`
Clock Domains	CamelCase with Clk suffix	`SystemClk`, `PeripheralClk`
Module Names	CamelCase	`MemoryController`, `VideoProcessor`
Parameters	ALL_CAPS	`CLOCK_FREQUENCY`, `FIFO_DEPTH`
Constants	ALL_CAPS with type prefix	`C_TIMEOUT_VALUE`, `N_BUFFER_SIZE`
Test Signals	CamelCase with test prefix	`test_ClockGen`, `test_DataValid`

3.2.2. Software Naming Conventions

NOTE Language-specific constraints may override these conventions. Each implementation chapter must document any deviations.

Table 4. Common Software Element Naming

Type	Preferred Convention	Examples	Language Exceptions
Function Names	camelCase	`getData()`, `writeBuffer()`	COBOL: ALL-CAPS Fortran: snake_case
Variables	snake_case	`buffer_size`, `current_state`	Pascal: PascalCase COBOL: ALL-CAPS
Constants	SCREAMING_SNAKE_CASE	`MAX_BUFFER_SIZE`	C++: kCamelCase
Class Names	PascalCase	`MemoryManager`	Generally consistent
File Names	kebab-case	`memory-controller.c`	See OS/Language constraints
Macros	SCREAMING_SNAKE_CASE	`ENABLE_DEBUG`	Generally consistent
Type Definitions	PascalCase with _t	`BufferState_t`	Language dependent

Table 5. Language-Specific Constraints

Language	Key Constraints
Pascal	Case-insensitive No underscores in identifiers Maximum identifier length varies by implementation
Forth	Case-sensitive Only ASCII characters allowed Short names preferred for stack efficiency
Modula-2	Case-sensitive Reserved words must be UPPERCASE Module names must match file names
Go	Case-sensitive First character determines visibility Package names must be lowercase
Python	Case-sensitive Indentation defines blocks PEP 8 style guide conventions
Lisp	Case-insensitive traditionally Hyphenated-names preferred Package/system specific conventions
6502 Assembly	Case-insensitive Label length limits vary by assembler Restricted first characters for labels

3.3. Visual Elements

3.3.1. Diagrams

Table 6. Diagram Requirements

Aspect	Specification
Format	PNG format, 300dpi minimum
Dimensions	Maximum 1920x1080 pixels
Storage	`images/` directory by section
Alt Text	Required for all diagrams
Source Files	Store Graphviz .dot files with PNG

Table 7. Graphviz Usage

Diagram Type	When to Use
Block Diagrams	System components and connections
Flow Charts	Processes and decision flows
State Machines	State transitions and conditions
Hierarchies	Organization and structure

3.3.2. Tables

Table 8. Table Formatting

Element	Requirements
Headers	Title case Bold text Clear, concise labels
Content	Left-align text Right-align numbers Consistent capitalization
Format	Use grid for complex data Use header row Consistent column widths

3.3.3. Code Blocks

Table 9. Code Block Requirements

Element	Requirements
Language	Specify for syntax highlighting
Line Numbers	Required for blocks > 10 lines
Indentation	Spaces, not tabs

Element	Requirements
Comments	Required for complex sections

3.4. Version Control

Table 10. Version Identifiers

Type	Format	Example
Hardware	vM.m.p	v2.1.3
Software	vM.m.p	v1.0.5
Documentation	vM.m.p-doc	v2.1.3-doc