

Aves Documentation Guide

Table of Contents

1. Introduction	1
2. Aves Documentation Plan	2
2.1. Introduction	2
2.2. Documentation Standards	2
2.3. Style Guide	3
3. Documentation Organization Plan	7
3.1. Document Structure Guidelines	7
3.2. Implementation Notes	10

Chapter 1. Introduction

This guide combines the complete documentation standards and organization for the Aves project.

Chapter 2. Aves Documentation Plan

2.1. 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.

2.2. Documentation Standards

2.2.1. Document Organization

Section Hierarchy

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

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

2.3. Style Guide

2.3.1. Writing Style

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

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

2.3.2. Naming Conventions

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

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

2.3.3. Visual Elements

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

Diagram Type	When to Use
Hierarchies	Organization and structure

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

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
Comments	Required for complex sections

2.3.4. 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

Chapter 3. Documentation Organization Plan

3.1. Document Structure Guidelines

3.1.1. Common Front Matter Components

Each manual will have the following structure with separate files:

- book-info.adoc (ISBN, copyright, publication info)
- title-page.adoc
- abstract.adoc
- foreword.adoc
- contents.adoc
- preface.adoc

3.1.2. Directory Structure

```
hardware-manual/  
├── front-matter/  
│   ├── book-info.adoc  
│   ├── title-page.adoc  
│   ├── abstract.adoc  
│   ├── foreword.adoc  
│   ├── contents.adoc  
│   └── preface.adoc  
├── parts/  
│   ├── part1-architecture/  
│   │   ├── chapter1-intro.adoc  
│   │   ├── chapter2-overview.adoc  
│   │   ├── chapter3-design.adoc  
│   │   ├── chapter4-computational-models.adoc  
│   │   └── chapter5-system-configurations.adoc  
│   ├── part2-components/  
│   │   ├── chapter1-overview.adoc  
│   │   ├── chapter2-modules.adoc  
│   │   ├── chapter3-interfaces.adoc  
│   │   └── chapter4-specifications.adoc  
│   └── part3-video/  
├── back-matter/  
│   ├── appendices.adoc  
│   ├── glossary.adoc  
│   └── index.adoc  
└── hardware-manual.adoc (main include file)
```

```

software-manual/          # Based on existing Kingfisher documentation
├── front-matter/
│   ├── book-info.adoc
│   ├── title-page.adoc
│   ├── abstract.adoc
│   ├── foreword.adoc
│   ├── contents.adoc
│   └── preface.adoc
├── parts/
│   ├── part1-getting-started/
│   │   ├── chapter1-introduction.adoc
│   │   ├── chapter2-quick-start.adoc
│   │   └── chapter3-basic-concepts.adoc
│   ├── part2-fundamentals/
│   │   ├── chapter1-architecture-overview.adoc
│   │   ├── chapter2-core-concepts.adoc
│   │   └── chapter3-basic-operations.adoc
│   ├── part3-architecture/
│   │   ├── chapter1-system-architecture.adoc
│   │   ├── chapter2-hardware-model-support.adoc
│   │   └── chapter3-implementation.adoc
│   ├── part4-runtime/
│   │   ├── chapter1-execution-model.adoc
│   │   ├── chapter2-memory-management.adoc
│   │   └── chapter3-io-subsystem.adoc
│   ├── part5-development/
│   │   ├── chapter1-development-environment.adoc
│   │   ├── chapter2-programming-guide.adoc
│   │   └── chapter3-debugging.adoc
│   └── part6-reference/
│       ├── chapter1-api-reference.adoc
│       ├── chapter2-system-calls.adoc
│       └── chapter3-hardware-interfaces.adoc
├── back-matter/
│   ├── appendices.adoc
│   ├── glossary.adoc
│   └── index.adoc
└── software-manual.adoc (main include file)

```

3.1.3. Main Document Structure

hardware-manual.adoc template

```
[existing hardware manual content]
```

software-manual.adoc template

```

= Aves Software Reference Manual
:doctype: book
:toc:

```

:sectnums:

```
Unresolved directive in aves-org-plan.adoc - include::front-matter/book-info.adoc[]
Unresolved directive in aves-org-plan.adoc - include::front-matter/title-page.adoc[]
Unresolved directive in aves-org-plan.adoc - include::front-matter/abstract.adoc[]
Unresolved directive in aves-org-plan.adoc - include::front-matter/foreword.adoc[]
Unresolved directive in aves-org-plan.adoc - include::front-matter/contents.adoc[]
Unresolved directive in aves-org-plan.adoc - include::front-matter/preface.adoc[]

Unresolved directive in aves-org-plan.adoc - include::parts/part1-getting-
started/*.adoc[]
Unresolved directive in aves-org-plan.adoc - include::parts/part2-
fundamentals/*.adoc[]
Unresolved directive in aves-org-plan.adoc - include::parts/part3-
architecture/*.adoc[]
Unresolved directive in aves-org-plan.adoc - include::parts/part4-runtime/*.adoc[]
Unresolved directive in aves-org-plan.adoc - include::parts/part5-development/*.adoc[]
Unresolved directive in aves-org-plan.adoc - include::parts/part6-reference/*.adoc[]

Unresolved directive in aves-org-plan.adoc - include::back-matter/appendices.adoc[]
Unresolved directive in aves-org-plan.adoc - include::back-matter/glossary.adoc[]
Unresolved directive in aves-org-plan.adoc - include::back-matter/index.adoc[]
```

3.1.4. Git Branch Strategy

Main Branch Structure

- **main** - stable, released version
- **develop** - integration branch

Feature Branch Organization

Feature branches should be organized by logical content groups:

feature/hw-core-architecture	# Architecture and computational models
feature/hw-subsystems	# Components and video sections
feature/sw-user-guide	# Getting started and fundamentals
feature/sw-technical-reference	# Architecture, runtime, and development

Task-Based Branches

For specific updates:

```
feature/front-matter-copyright-update
feature/hw-diagrams-update
feature/sw-api-documentation
feature/common-glossary-update
```

3.1.5. Build Configuration

Makefile targets for different outputs:

```
pdf-full: Generate complete PDF with all sections
pdf-front: Generate front matter only (for review)
pdf-body: Generate main content without front/back matter
pdf-parts: Generate individual part PDFs
html: Generate HTML version
```

3.2. Implementation Notes

1. Front Matter

- Each component in separate file for independent review/printing
- Consistent styling across both manuals
- Version control friendly
- Maintained in main branch structure, not separate branch

2. Main Content

- Organized by parts and chapters
- Cross-references use document IDs
- Images in separate directories
- Code listings as separate files

3. Build System

- Makefile targets for different outputs
- PDF generation with proper page numbering
- HTML with navigation
- Index generation
- Cross-reference validation

4. Version Control

- Each part can be worked on independently
- Clear separation of content and structure
- Easy to track changes in specific sections
- Regular merges from develop to feature branches

5. Quality Control

- Separate build targets for reviewing sections
- Validation of cross-references
- Style guide compliance checking
- Technical review process

- Branch protection on main and develop

6. Migration Notes

- Software manual based on existing Kingfisher documentation
- Restructure Kingfisher content to match new organization
- Preserve existing content while reorganizing
- Update cross-references and links
- Ensure consistent styling with hardware manual