

The Birds Project

The libBirds Library, Software Accomplishment Summary

Ronald S. Burkey

Version 0.5

10/31/01

Copyright © 2001 by Ronald S. Burkey

Licensing information: Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation with Invariant Sections "GNU Free Documentation License" and "GNU Lesser General Public License", with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

1. Purpose of the SAS Document

This is a standard "Software Accomplishment Summary" document, corresponding to the guidelines in RTCA DO-178B. It includes overviews of the system and its software, certification considerations, software characteristics, life-cycle information, and the change-history of the software.

This document is available for free use, under the terms of the GNU Free Documentation License (FDL). The text of the FDL is reproduced later in this document.

2. System Overview

2.1. Overview of the System

The libBirds library is intended to be a library of code which has been pre-certified under DO-178B, and which is therefore available for use as "previously written software" in building airborne-software applications. In other words, libBirds is not a complete system, but can be used as a software component of an airborne system without further development or certification effort.

The libBirds library is available for free use by anyone, under the terms of the GNU Lesser General Public License (LGPL). The associated documentation (such as the DO-178B documents) is also available for free use, under the terms of the GNU Free Documentation License (FDL). The text of both licenses appears later in this document.

The libBirds library attempts, philosophically, to meet the following additional criteria:

- 1) To supply a minimal C-callable API appropriate for writing simple embedded airborne applications, such as file-system functions, string manipulation functions, timekeeping functions, and so forth.
- 2) To supply only the most reliable types of functions. For example, heap (memory-allocation) operations are omitted, functions such as `strcpy` are omitted in favor of `memcpy`, functions such as `fgets` are omitted in favor of `fread`, and so on.
- 3) To retain maximal portability from CPU type to CPU type is desired, with a minimum porting effort. If hardware is designed with the requirements of existing libBirds code in mind -- i.e., with supported hardware peripherals -- then no effort at all is needed.

2.2. System Functions

2.2.1. Fixed-Size Datatypes for External Operations

2.2.1.1. Description

The library shall provide a set of fixed-size integer datatypes (i.e., in which the datatypes consist of known numbers of bits), so that external interfaces and the internal characteristics of the file-system can be dealt with on a consistent basis. Application software is free in most instances to use the normal C/C++ integer datatypes such as `int`, `long int`, `unsigned int`, and so on, but should use the fixed-size datatypes when accessing files or hardware interfaces, to insure maximum portability.

Fixed-size datatypes supported shall include 8-bit signed and unsigned integers, 16-bit signed and unsigned integers, 32-bit signed and unsigned integers, and 64-bit signed and unsigned integers.

2.2.1.2. Hardware-Software Allocation

This functionality is implemented in software.

2.2.2. Endian Conversions

2.2.2.1. Description

Because the aim of the libBirds library is to provide maximal portability, and yet byte-ordering within multi-byte datatypes differs from CPU type to CPU type, byte-ordering conversion functions are provided. For most coding purposes, byte-ordering preferences of the CPU are unknown and irrelevant, but they become relevant in file operations and when accessing multi-byte hardware interfaces. Therefore, application software should employ the appropriate byte-ordering conversion functions in these circumstances.

The libBirds library shall provide the following functionality: conversion (in either direction) between the CPU's native 16-bit or 32-bit integer datatypes, and 16-bit or 32-bit big-endian integer datatypes. Also, between the CPU's native 16-bit or 32-bit integer datatypes and 16-bit or 32-bit little-endian integer datatypes.

2.2.2.2. Hardware-Software Allocation

This functionality is implemented entirely in software.

2.2.3. Timekeeping Functions

2.2.3.1. Description

The libBirds library shall provide, at a minimum, the ability to determine the amount of time which has passed since system power-up. The granularity of the time measurement is not significant in libBirds, but for design purposes can be regarded as being of the order of magnitude of 10 ms. or smaller.

2.2.3.2. Hardware-Software Allocation

This shall be implemented by means of an interrupt service routine based on a hardware timer, such as an integrated CPU timer.

However, only the interrupt-service routine itself shall be provided by libBirds. The low-level details of setting up the interrupt and vectoring to the ISR are handled by a board-support package (not a part of libBirds), and hence this hardware dependence is transparent to libBirds.

2.2.4. String and Memory Manipulations

2.2.4.1. Description

Operations shall be provided such as comparing or copying strings or memory, case conversions, and so on. The extent of this functionality is not specified in the Requirements Process.

2.2.4.2. Hardware-Software Allocation

This functionality shall be implemented entirely in software.

2.2.5. Text-Display Functions

2.2.5.1. Description

A set of functions shall be provided appropriate for outputting data to a text-oriented display. By a "text-oriented" display is meant a display screen considered as rows and columns of characters, and not accessible other than at integral character cells. This concept is not dependent on the display hardware actually being text-oriented at a hardware level, of course, since a graphical display can also logically be considered as a text-oriented display.

The types of functions provided by libBirds shall include (but not necessarily be limited to) positioning the cursor at arbitrary text cells and displaying a character or string at a given text cell.

Arbitrary 24-bit color mappings shall be provided at the software level, though not necessarily at the hardware level.

2.2.5.2. Hardware-Software Allocation

The implementation of this functionality will vary depending on whether the actual display screen is text-oriented or graphically-oriented in hardware. The primary implementation difficulty is the conversion of character data to pixel data.

In the former case, this functionality can be completely provided by a board-support package (separate from libBirds). In the latter case, libBirds must break text operations down into pixel-manipulations, and the board-support package must provide the raw pixel manipulations.

Thus, libBirds is aware of hardware dependence only to the extent of knowing whether the display hardware is text-oriented or pixel-oriented.

2.2.6. Graphical Display-Output Functions

2.2.6.1. Description

A set of functions shall be provided allowing output to a graphically-oriented display screen.

The functionality provided shall include, but not necessarily be limited to: the ability to output text in various fonts and sizes to arbitrary pixel locations; the ability to draw

arbitrary lines or filled areas; the ability to display arbitrary graphics files in BMP format. Both aliased and non-aliased fonts shall be provided.

At a software level, 24-bit colors shall be provided, whatever the actual capabilities of the hardware display.

2.2.6.2. Hardware-Software Allocation

This functionality is provided entirely in software, except that the hardware display must have the capability of manipulating arbitrary pixels. The raw-pixel manipulations are provided by a board-support package (separate from libBirds).

2.2.7. Keyboard-Input Functions

2.2.7.1. Description

Keyboard data is provided to the application software by means of a FIFO buffer and associated functions for adding/removing data to/from the buffer. Separate buffer events shall be generated for key-depression and for key-release.

Keystrokes shall be debounced prior to their press/release events being added to the keyboard FIFO.

2.2.7.2. Hardware-Software Allocation

An interrupt-service routine shall scan the keyboard with some (unspecified) degree of regularity, debounce the keystrokes, and insert key-pressed and key-released events into the keyboard buffer.

The physical keyboard, of course, shall be in hardware.

Low-level details of setting up the interrupts, vectoring to the ISR, and fetching raw data from the physical keyboard are handled not by libBirds, but by a board-support package (separate from libBirds).

2.2.8. Filesystem Manipulations

2.2.8.1. Description

The ability to create or read "files" shall be present. The file-system is intended to be general-purpose, but limited in ways appropriate to embedded systems and to the use of flash-memory as a storage medium.

In particular, the following characteristics of familiar filesystems in Microsoft Windows or UNIX shall be provided: sub-directory structures; long filenames; files up to 2 Gbytes in size; reading and writing files sequentially; reading files randomly.

The following familiar filesystem characteristics shall not be supported: random writing; timestamps; ownership; permissions; sharing.

The filesystem software shall not be reentrant.

Files may be deleted, but their allocated space is not necessarily immediately reclaimed. The reclamation may require "garbage collection", as described in the next section.

2.2.8.2. Hardware-Software Allocation

The design criteria assume the use of flash-memory as a storage medium. In other words, the hardware medium must have the following properties: It may be erased in relatively large "erasable blocks", and erasure consists of setting the block to bits that are all 1; any bit which is 1 may be changed at will to 0, but not necessarily vice-versa. Of course, a file-system based on these limitations may also be implemented in other media (EEPROM, RAM, or magnetic disk), as well as flash-memory, though not with maximum efficiency.

All other capabilities are implemented in software.

In accordance with common practice, erasable blocks are sub-divided into smaller blocks ("sectors").

The following low-level functions are to be provided by a board-support package (separate from libBirds): mapping of the erasable blocks by address and size; erasure of blocks; reading sectors; writing sectors.

2.2.9. Filesystem Garbage Collection

2.2.9.1. Description

Because the file-system is conceptually based on flash-memory as a storage medium, as described above, space which has been used but subsequently deallocated cannot be immediately reclaimed. This is because bits which have been changed from 1 to 0 cannot be returned to 1 unless the entire block containing them is erased.

The libBirds library shall provide a garbage collection facility capable of reclaiming used file-system space by temporarily buffering a block of data within RAM while the associated flash-memory is being erased, and then writing the buffered data back to flash-memory afterward with reclaimed 0-bits changed to 1-bits as appropriate.

2.2.9.2. Hardware-Software Allocation

The hardware-software allocation for this functionality is the same as described above under "Filesystem Manipulations".

2.2.10. Audio Playback

2.2.10.1. Description

Audio clips stored within the file-system as files in the WAV format may be played back through an audio codec, if one exists.

2.2.10.2. Hardware-Software Allocation

A compatible audio codec must exist within hardware to use this feature. An interrupt-service routine is used to transfer audio data from the file-system to the audio codec.

The low-level details of setting up the interrupts, vectoring to the ISR, setting up the codec, and outputting data to the codec are handled by a board-support package (separate from libBirds).

2.2.11. Serial I/O

2.2.11.1. Description

The libBirds library shall provide functions that can be used for simple serial i/o. "Simple" serial i/o is defined as i/o involving only 8-bit data without parity or handshaking, and with loose (or no) timing constraints. This functionality is provided by means of FIFO buffers into which serial data is placed or removed.

This same mechanism shall also be used if ethernet or other network services are provided.

2.2.11.2. Hardware-Software Allocation

Providing this functionality is based on the existence of hardware UARTs, and an interrupt-service routine servicing these UARTs. The interrupt-service routine receives data from the UARTs and places the data into a "received data" FIFO from which the application software can remove it. Similarly, the application software can place data into a "transmitter" FIFO from which the interrupt-service return removes it and gives it to the UARTs.

Low-level details of setting up the interrupts, vectoring to the ISR, and inputting/outputting data from/to the UARTs is handled by a board-support package (separate from libBirds).

2.3. System Architecture

Because the libBirds library is intended to be as portable as possible, it requires no specific system architecture.

For example, no specific requirements on the quantity or address range of RAM is made. It is assumed, though not required, that the total system RAM is greater than 128K bytes. Where specific system functions have significant memory requirements relative to 128K bytes, it is stated in the design data.

Also, certain functionality (such as audio playback) requires the availability of compatible hardware (such as an audio codec). In no case does libBirds have specific knowledge of these hardware peripherals or access to them. Instead, all such access is via functions provided by a board-support package (separate from libBirds). The board-support functions needed are specified in the design documentation.

2.4. Processors

The libBirds library can be used with any CPU supported by the GNU **gcc** C/C++ compiler. For practical purposes, this means that almost any popular 32-bit CPU is supported.

2.5. Hardware-Software Interfaces

As may be deduced from the descriptions of the hardware-software allocations in the 'System Overview' section, libBirds as such has almost no dependence on or knowledge of the hardware.

Instead, there is a hardware-abstraction layer (HAL), whose functions are defined in the libBirds design data, but which are actually provided by "board-support packages" (BSPs). The BSPs are separate from libBirds, and have life cycles separate from libBirds.

Since libBirds provides specifications for all HAL functions, as well as the test suites used for software verification of the BSPs, it is hoped that certification of the BSPs for given versions of libBirds will be a comparatively simple effort.

2.6. Safety Features

Because libBirds is a reusable library rather than a complete system, it does not attempt to provide safety features as such. The libBirds library promotes safety primarily in omitting functionality that can easily be misused to the detriment of system reliability, such as dynamic memory allocation.

2.7. Differences from PSAC

There are no differences between the System Overview as described in the PSAC and as described in the SAS.

3. Software Overview

3.1. Partitioning

TBD

3.2. Resource Sharing

The libBirds library is intended to be used in a single-tasking system without multiple CPUs or (if multiple CPUs are present) without shared memory. It uses no globally accessible memory, other than memory allocated specifically for the use of libBirds, and not used by other (properly designed) software. It shares no communications channels. It expects compatible peripherals to be accessed only by means of libBirds functions. The libBirds library provides no functions capable of retaining permanent control of the CPU.

Thus, the only applicable questions about resource sharing are the proportion of total RAM and total CPU time used, and the maximum execution time required by a libBirds function. RAM usage should be deduced from the specifically required system functions by referring to the design data.

Unfortunately, questions about CPU utilization cannot be answered since libBirds specifies no specific CPU or clock speed. Hence, it should be concluded that libBirds is not suitable for use in applications where hard limits CPU utilization are needed.

3.3. Redundancy

Not relevant, since libBirds is a reusable library rather than a complete system.

3.4. Multiple-Version Dissimilar Software

Not used by libBirds.

3.5. Fault-Tolerance, Failure Detection, and Safety Monitoring

Not provided by the libBirds library.

3.6. Software Timing and Scheduling Strategies

The libBirds library assumes that libBirds functions and all other code for the system (such as application code) run in a single foreground execution thread. Thus, application code passes control to libBirds functions when it wants to do so, and receives control back when the libBirds function has terminated.

However, libBirds functions can only provide certain functionality by depending on some underlying interrupt-service routines, as follows:

- 1) The libBirds "kernel" ISR executes at regular intervals by means of a CPU timer or other hardware timer. This ISR handles the following tasks: updating the master system clock; scanning/debouncing the keyboard; transferring audio data from the file-system to the audio codec; user-defined operations via a function call reserved for this purpose.
- 2) Interrupt-service routines for each supported UART.

While libBirds provides the ISR, low-level details of setting up the hardware (interrupts, timers, UARTs) and vectoring to the ISR are handled not by libBirds, but by a board-support package (separate from libBirds).

3.7. Differences from PSAC

There are no differences between the Software Overview as described in the PSAC and as described in the SAS.

4. Certification Considerations

4.1. Software Level and Means of Compliance

The software is suitable for certification via RTCA DO-178B at level C.

4.2. Justification of Software Level

Since libBirds is a reusable library, rather than a complete system, it requires no safety justification as such.

4.3. Potential Software Contributions to Failure Conditions

Because the conditions of use of libBirds cannot be known (it may be used in developing software for any software of level C, D, or E), there is no way to know how libBirds may potentially contribute to failure conditions. In other words, there is no justification for using libBirds in developing software at levels A or B.

4.4. Differences from PSAC

There are no differences between the Certification Considerations as described in the PSAC and as described in the SAS.

5. Software Characteristics and Constraints

5.1. Executable Object Code Size

5.1.1. Strategy for Management

TBD

5.1.2. Measured Value

TBD

5.2. Timing Requirements and Constraints

5.2.1. Strategy for Management

TBD

5.2.2. Measured Value

TBD

5.3. Memory Size Constraints

5.3.1. Strategy for Management

TBD

5.3.2. Measured Value

TBD

5.4. TBD Name of additional limited system resource

5.4.1. Strategy for Measurement

TBD

5.4.2. Measured Value

TBD

6. Software-Component Life Cycles

For this project, there is only one software component, namely the libBirds library, and hence only one life cycle.

6.1. Life Cycle of libBirds Library Development

6.1.1. Life-Cycle

The life cycle of the libBirds library begins with the Planning Process. Upon the end of the planning processes, three separate chains of development begin, and these chains persist until the end of the development effort.

Two of the of the development chains consist of one process each: namely, the SCM Process and the SQA Process.

The third development chain consists of 4 processes: the Requirements Process, followed by the Design Process, the Coding Process, the Integration Process, and the Software Verification Process. For simplicity, we'll refer to this as the "RDCIV chain." The development effort basically progresses through these processes, in the order given, but can backtrack to an earlier process upon discovery of errors that can only be corrected in the earlier process.

Upon completion of the development effort, which is the release of the software by the SQA Process, life cycle data is available for input to a Certification Liaison Process. However, the Certification Liaison Process is really outside of the scope of the libBirds development effort, since the software produced is merely a reusable library and not a complete system.

Although not possible for the first libBirds release, due to non-availability of personnel, it is hoped that subsequent releases can be reviewed by DER as part of the development effort, allowing very rapid signoff of form 8110 for developers using the libBirds library. When this becomes possible, it will form part of the Certification Liaison Process.

Figure 1. Life Cycle Summary

6.1.1.1. Differences from the PSAC

There are no differences between the life cycle as described in the PSAC and as described in the SAS.

6.1.2. Life-Cycle Data

6.1.2.1. Data Items

Because libBirds is a reusable library rather than a complete airborne product, no occasion for submission of life cycle data by the Birds Project arises. Rather, all of the life cycle data items described below are made available to developers wishing to use the libBirds library, or to anyone else, along with libBirds source code. The status of this data within the development efforts of libBirds users cannot in principle be known to the libBirds development effort, and hence cannot be specified here.

The following items or categories of life cycle data items are created.

The Plan for Software Aspects of Certification (PSAC) is created by the Planning Process, but the 'System functions' section is provided by (and hence the PSAC is completed by) the Requirements Process.

The Software Development Plan (SDP) is created by the Planning Process.

The Software Verification Plan (SVP) is created by the Planning Process.

The Software Configuration Management Plan (SCMP) is created by the Planning Process.

The Software Quality Assurance Plan (SQAP) is created by the Planning Process.

The Software Requirements Standards (SRS) are created by the Planning Process.

The Software Design Standards (SDS) are created by the Planning Process.

The Software Code Standards (SCS) are created by the Planning Process.

The Software Requirements Data (SRD) are created by the Requirements Process.

The Software Design Description (SDD) is created by the Design Process.

The Source Code is created by the Coding Process.

There is no Executable Object Code in general, since libBirds is a reusable library rather than a complete product. Creation of Executable Object Code is performed by board-support package development efforts (separate from libBirds). For specific CPU types used in the libBirds Software Verification Process, however, Executable Object Code in the form of linkable libraries can be provided.

The Software Verification Cases and Procedures (SVCP) and Software Verification Results (SVR) are created or completed by the Software Verification Process.

The Software Life Cycle Environment Configuration Index (SECI) is created by the Planning Process, but may be altered by the SCM Process prior to creation of the Executable Object Code (if any) or transition to the Software Verification Process.

The Software Configuration Index (SCI) is produced by the SCM Process.

Problem Reports are sanctioned and maintained by the SQA Process, but may actually be produced at any time, by anyone with access to the libBirds problem-reporting system.

SCM Records are produced by the SCM Process.

SQA Records are produced by the SQA Process.

The Software Accomplishment Summary (SAS) is produced by the SQA Process.

6.1.2.2. Data Relationships

All relationships among life cycle data items seem clear from the 'Life-Cycle' section above.

6.1.2.3. Means of Submitting Life-Cycle Data

Since libBirds is a reusable library rather than a complete product, no direct submittal of data by the Birds Project is envisaged. However, all life cycle data (code and documentation) are available to software developers, certification authorities, or any other parties, via download over the Internet.

In practice, it is envisaged that developers wishing to use libBirds for their projects will download the source code and other life cycle data, will archive them within their own SCM Processes, and will perform whatever submissions are required.

6.1.2.4. Differences from the PSAC

TBD

7. Additional Considerations

7.1. Alternate Methods of Compliance

No qualification means alternate to DO-178B are used.

7.2. Tool Qualification

In general, a tool requires qualification if its output is used without examination. If the output of the tool is itself verified (by review, testing, or analysis) this constitutes an implicit qualification of the tool itself. With that in mind, we can examine the tools used, one-by-one, and determine their need for qualification:

Native compiler, linker, library archiver, and low-level libraries (**gcc**, **ld**, **ar**, and **libgcc.a**). These are used for creating desktop-computer executable code for testing purposes, or for creating embedded code if the target processor just happens to be the same as that of a common desktop computer (like Intel 'x86 or PowerPC). These tools do not require qualification, since their outputs (the executable code) are tested.

Cross-compiler, linker, library archiver, and low-level libraries (**gcc**, **ld**, **ar**, and **libgcc.a**). These are used for creating embedded Executable Object Code from a desktop computer, but for a different target CPU type. Technically, these tools also do not need qualification, since libBirds does not produce Executable Object Code for these environments as part of its life cycle data. Rather, separate board-support package development efforts create this Executable Object Code. [As a practical matter, however, the Birds Project wants the libBirds library to be as easily usable by developers as possible, and this means that this issue cannot be side-stepped. Easing

BSP qualification is handled by crafting the test suites so that they can be executed in either the desktop environment or in the target environment (albeit with perhaps more effort).]

Coverage tool (**gcov**). This is a tool which provides a survey of all source-code lines, indicating which of them have been exercised in testing and which have not. The coverage tool must be qualified, since there is no direct way of verifying its output. The qualification shall be done by creating a C-language program in which various bits of code are known to execute or not execute, and then to test it with **gcov**.

Code-formatting tool (**indent**). This tool puts the C-language source code into a standard format. Although the outputs from **indent** are not examined, it is still not necessary to qualify the tool. This is because the SCS does not place a requirement on the format of the output code, but simply requires that **indent** has been run. Therefore, whatever output 'indent' produces is correct by definition.

Concurrent versioning system (**cvs**). This is the archiver used for the Source Code and other life cycle data. It is qualified by creating a suite of data that must be archived, and then retrieved and compared against the original data. In addition to this qualification, part of the release procedure is to retrieve the archived Source Code and check it for accuracy. Unfortunately, the latter step alone is not adequate to bypass qualification, since the qualification process needs to insure not merely that the current release can be retrieved, but that prior releases can be retrieved as well.

All other tools used either have outputs which are examined, or which feed into operations whose outputs are examined. Therefore, no other tools require qualification.

7.3. Previously-Developed Software

Previously-developed software is not used by libBirds.

7.4. Option-Selectable Software

The libBirds library is not, and does not contain, option-selectable software.

7.5. User-Modifiable Software

The libBirds library is not, and does not contain, user-modifiable software.

7.6. Commercial Off-the-Shelf Software

The libBirds library does not use COTS.

7.7. Field-Loadable Software

The libBirds library is not field-loadable as such, but could conceivably be used by a developer producing a field-loadable program. If so, any considerations relating to this will be outlined in that developer's life cycle data.

7.8. Multiple-Version Dissimilar Software

The libBirds library does not use multiple-version dissimilar software.

7.9. Product Service-History

Not applicable.

7.10. Issue Papers

TBD

7.11. Special Conditions

TBD

7.12. TBD title of an additional SAS consideration

TBD

8. Software Identification

8.1. TBD version code of a specific release

8.1.1. Change History

8.1.1.1. Changes from the Prior Release

8.1.1.1.1. TBD specific change

Traceability: SAS section 8.1.1.1.1 → SVCP section 5.1.

8.1.1.1.1.1. Description

TBD

8.1.1.2. Unresolved Problem Reports

8.1.1.2.1. TBD reference to specific problem report

8.1.1.2.1.1. Brief Description

TBD

8.1.1.2.1.2. Functional Limitations

TBD

8.1.1.2.1.3. Safety Justification

TBD

9. Statement of Compliance

TBD

10. GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or

noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions

of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five). C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released

under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.