The Birds Project

The libBirds Library, Plan for Software Aspects of Certification

Ronald S. Burkey

Version 0.5
10/31/01

1. Purpose of the PSAC Document

This is a standard "Plan for Software Aspects of Certification" document, corresponding to the guidelines in RTCA DO-178B. It describes the general characteristics of the system and its software, certification considerations, life cycles and life-cycle data, and scheduling of the software-development effort.

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 be 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.

3. Software Overview

3.1. 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.

3.2. Redundancy

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

3.3. Multiple-Version Dissimilar Software

Not used by libBirds.

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

Not provided by the libBirds library.

3.5. 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).

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.

5. Software-Component Life Cycles

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

5.1. Life Cycle of libBirds Library Development

5.1.1. 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.

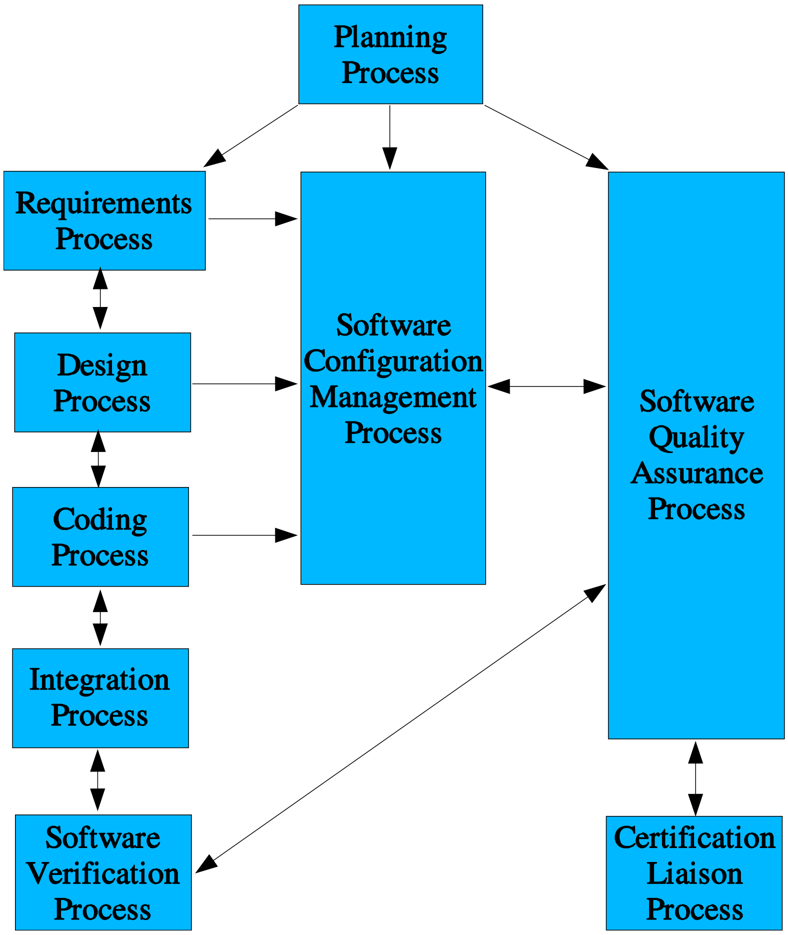
Two of the of the development chains consist of one process each: namely, the SCM Process and the SQA Process.

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



5.1.1.1. Life-Cycle Processes

5.1.1.1.1. Planning Process

The Planning Process precedes all other life-cycle processes It produces or identifies all other Plans or Standards guiding the remainder of the software-development effort. The specific aim of the libBirds Planning Process is to address all of the issues outlined in DO-178B section 4.0.

5.1.1.1.1.1. Transition Criteria and Satisfaction of Objectives

The Planning Process is followed by three separate chains of life-cycle processes, with the three development chains running simultaneously in parallel. These chains are the SCM Process, the SQA Process, and the RDCIV chain (see the ’Life-Cycle’ section). The transitions from the Planning Process to these development chains do not necessarily occur simultaneously.

The Planning Process transitions to the RDCIV chain when the PSAC, SDP, SVP, SECI, SRS, SDS, and SCS documents have all been successful reviewed and signed off. Note that the PSAC, unlike the other documents mentioned, is not actually completed at this point since it contains some high-level requirements (notably under the heading of ’System Overview’) that are not known until the Requirements Process.

Subsequent changes to these documents may require a return from the Design Process, Coding Process, or Software Verification Process to the Requirements Process, but the Planning Process is never re-entered in any given life cycle.

The Planning Process transitions to the SCM Process upon successful review and signoff of the SCMP.

The Planning Process transitions to the SQA Process upon successful review and signoff of the SQAP.

5.1.1.1.2. Requirements Process

The purpose of the Requirements Process is to develop the high-level software requirements. In the case of libBirds, this is essentially a development of the functional requirements. However, the intention of the libBirds Requirements Process is to address all of the issues outlined in DO-178B section 5.1.

The Requirements Process also addresses all considerations of DO-178B section 6.3.1, which according to DO-178B may form a part of the Software Verification Process. This can be done because at the proposed software level (’C’), there is no requirement of independence.

5.1.1.1.2.1. Transition Criteria and Satisfaction of Objectives

The Requirements Process transitions to the Design Process upon successful review and signoff of the SRD. The completion of the incomplete PSAC produced in the Planning Process is also needed, along with its review and signoff.

5.1.1.1.3. Design Process

In the case of libBirds, the main purpose of the Design Process is to develop low-level requirements (primarily API definition) from the high-level requirements provided by the Requirements Process. However, all of the issues outlined in DO-178B section 5.2 are addressed.

The Requirements Process also addresses all considerations of DO-178B sections 6.3.2 & 6.3.3, which according to DO-178B may form a part of the Software Verification Process. This can be done because at the proposed software level ('C'), there is no requirement of independence.

5.1.1.1.3.1. Transition Criteria and Satisfaction of Objectives

The Design Process transitions to the Coding Process upon successful review and signoff of the SDD.

5.1.1.1.4. Coding Process

The purpose of the Coding Process is to produce source code and object code conforming to the SDD, SDS, and SCS. It addresses the issues outlined in DO-178B section 5.3.

5.1.1.1.4.1. Transition Criteria and Satisfaction of Objectives

The Coding Process Transitions to the Integration Process when the source code and object code are prepared, have been verified conformant with the SCS and SDS, and have been verified conformant with (and traceable to) the SDD.

5.1.1.1.5. Integration Process

The purpose of the Integration Process is to integrate the software with the target hardware. Since the aim of the libBirds project is to produce a highly portable library, rather than a hardware-specific library or a physical device, there really can be no required Integration Process.

For any given target architecture, the general-purpose libBirds library is combined with a specific 'board-support package' (not a part of libBirds as such) that provides the Hardware Abstraction Layer specific to that target. It is in the Integration Process of the board-support package life cycle or the application code life cycle (both of which are independent of libBirds) that the integration occurs.

5.1.1.1.5.1. Transition Criteria and Satisfaction of Objectives

At the option of the developers, the Integration Process may either proceed directly to the Software Verification Process without any effort whatever, or else to optionally coordinate with board-support package (BSP) development-effort Integration Processes. There are no pre-defined criteria associated with this decision.

5.1.1.1.6. Software Verification Process

In the words of DO-178B section 6.1, "The purpose of the software verification process is to detect and report errors that may have been introduced during the software development processes." The libBirds Software Verification Process attempts to address all of the issues outlined in DO-178B chapter 6, except the following:

1) The Requirements Process addresses all considerations of DO-178B section 6.3.1. This can be done because at the proposed software level (’C’), there is no requirement of independence.

2) The Design Process addresses all considerations of DO-178B sections 6.3.2 & 6.3.3. This can be done because at the proposed software level (’C’), there is no requirement of independence.

3) Reviews and analysis of the outputs of the Integration Process (DO-178B section 6.3.5) are not addressed here (or elsewhere) since the entire Integration Process is optional. Refer to the ’Integration Process’ section of the PSAC for further explanation.

5.1.1.1.6.1. Transition Criteria and Satisfaction of Objectives

The Software Verification Process has several outputs:

a) Review and analysis of the source code.

b) The SVCP.

c) Review and analysis of the test results.

d) The software-test results themselves.

The Software Verification Process can transition to various other life cycle processes:

1) To the SQA Process upon successfully creating all Software Verfication Process outputs.

2) To the Coding Process upon detection of errors in software testing.

3) To the Requirements Process or Design Process upon detection of errors more appropriately resolved in the SRD or SDD than in the code.

5.1.1.1.7. Software Configuration Management Process

The Software Configuration Management Process (or just ’SCM Process’) for the most part operates simultaneously with the other life cycle processes. DO-178B chapter 7 sets out the objectives and activities of the SCM Process in some detail. In summary, the SCM Process provices the following activities:

1) Identifying configurations.

2) Implementing change control.

3) Establishing baselines.

4) Archiving the software and the life cycle data.

Though for graphical reasons the figure at the top of the ’Life-Cycle’ section of this document depicts the SCM Process as spanning merely the Requirements, Design, and Coding Processes, it actually spans all other life cycle processes, and beyond. Once data is archived by means of the SCM Process, it is theoretically intended to remain archived as long as the libBirds software is present within any airborne units.

Of course, since the Birds Project is not a manufacturer of airborne equipment, and the libBirds library is intended to be used by manufacturers of such equipment, it is not really within the capability of the Birds Project to guarantee this essentially indefinite data retention. It is therefore assumed that users of the libBirds library have prudently archived the version of libBirds they are using -- i.e., all code and life-cycle data -- within the SCM Processes of their own development efforts.

5.1.1.1.7.1. Transition Criteria and Satisfaction of Objectives

The SCM Process does not transition to other life cycle processes, since it operates in parallel with such other processes.

Ultimately, the SCM Process exists to archive life cycle data, and to perpetually maintain this archive subject to the limitations described above. The outputs of the SCM Process are the SCI and the SCM Records demonstrating archival activity. The SECI, which may legitimately be viewed as an output of the SCM Process, is actually produced by the Planning Process, but may subsequently be altered by the SCM Process.

5.1.1.1.8. Software Quality Assurance Process

The objectives and activities of the Software Quality Assurance Process (or just ’SQA Process’) are set out in chapter 8 of DO-178B. Basically, the SQA Process examines the outputs of the other life cycle processes and determines their internal consistency. In other words, it acts to assure that what has actually been accomplished is what was required to be accomplished.

Among the important activities of the SQA Process are these:

1) Establishement and management of the problem-reporting system.

2) Final release of the software.

The SQA Process operates simultaneously with the other life cycle processes, and spans the Planning Process through release of the software. Furthermore, the SQA Process is independent from the other life cycle processes, in the sense that separate personnel are involved and that the authority for the SQA Process is separate from the authority for the the other life cycle processes.

The SQA Process is the only life-cycle process required to have this characteristic of independence at the proposed software level (’C’).

5.1.1.1.8.1. Transition Criteria and Satisfaction of Objectives

The SQA Process transitions to the Certification Liaison Process upon release of the software. The outputs of the SQA Process are the SQA Records.

The primary output of the SQA Process is the Software Conformity Review, which is the last step in the release of the software. The Software Conformity Review, however, takes into account not only the life cycle data in general, but also various other SQA Records.

5.1.1.1.9. Certification Liaison

Since the product of the libBirds development effort is a reusable library rather than an actual airborne device, there are really no certification efforts associated with it, and thus no real Certification Liaison Process.

However, since the intention of the libBirds project is to not merely provide software but to make it very conveniently usable by developers, a very useful certification activity would be review and approval by a DER. This "pre-approval" by a DER would allow immediate signoff of 8110 forms for developers using the libBirds library, very simply and relatively cheaply.

It is not known as this is written whether such DER activity will actually occur. Hence, it should be regarded as an optional activity that may be omitted.

5.1.1.1.9.1. Transition Criteria and Satisfaction of Objectives

The outputs of the Certification Liaison Process can be these:

1) Informal notification that a DER finds the life cycle data acceptable and will be available to sign off (via 8110) upon demand, for a known fee.

2) Notification of life-cycle data problems that will require repair before the DER finds the data acceptable.

Upon the former (option #1), the Certification Liaison Process ends, but there are no further processes to which a transition can occur. Of course, the availability of signoff would be published, so that libBirds users could be made aware of it.

Upon the latter (option #2), problem reports are filed concerning the problems which have been found. Also, a transition may (or may not) occur to an earlier life cycle process so that the problems can be fixed. The reason for this uncertainty of action is that while approval by the DER is a desirable result of the development effort, it is not a crucially necessary one from the point of view of the ’Birds Project’. These findings are often a matter of opinion, and a different DER might view the life cycle data differently.

5.1.1.2. Organization

The Birds Project differs from most organizations involved in airborne software development, in the sense that it is not a commercial organization engaged in manufacturing a product. Instead, it is an effort to freely provide certification-friendly materials to the aviation community, as a service. Furthermore, at least at the time of inception of the libBirds development effort, it is not really "organized" at all, consisting merely of an ad hoc assemblage of a few individuals.

For this reason, the organization of the libBirds development effort cannot be understood in terms of an "org chart" with a neatly defined flow of authority. Instead, it can only be understood in terms of the actual individuals involved.

5.1.1.3. Organizational Responsibilities

At the proposed software level (’C’), the minimum number of individuals required for the development effort is two, to meet the ’independence’ requirements of DO-178B

Table A-9: one individual for the SQA Process, and one individual for all other activities. If the optional Certification Liaison Process is undertaken (see the ’Certification Liaison’ sub-section of ’Software Component Life Cycles’ in the PSAC document), then a third individual (a DER) is required.

For the initial release of libBirds, only the minimum number of personnel is used. For later releases, more or other personnel may be used. In this section, all personnel involved in all releases are identified, and their levels of involvement in these releases are made clear. Because there is no corporate entity providing authority for the development activities, we also provide brief resumes of participating individuals, so that their qualifications for their activities are clear.

Ronald S. Burkey, lead developer for libBirds v1.00, has a B.S. degree in Mathematics and a Ph.D. in Physics. He has been professionally involved in designing electronic hardware and firmware for airborne applications (and non-airborne applications) since 1984. He has been responsible for both DO-178A and DO-178B certification efforts.

TBD, SQA Manager for libBirds v1.00.

5.1.1.4. Certification Liaison

Because libBirds is a reusable library rather than a complete airborne product, no direct interaction with certification authorities is required.

5.1.2. Life-Cycle Data

5.1.2.1. 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.

5.1.2.2. Data Relationships

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

5.1.2.3. Data Formats

All data items other than Source Code and Executable Object Code are made available as Adobe PDF files, viewable with the freely available Adobe Acrobat Reader program. This includes the SVR, and all SCM Records and SQA Records (such as review and audit forms). If necessary, hand-written data is scanned in order to produce the necessary PDF files.

Source Code is made available as a UNIX ’tar’ file, compressed with the ’gzip’ utility. Executable Object Code (actually, linkable libraries) for the specific CPU types used in the Software Verification Process UNIX-type ’ar’ libraries.

Though not specifically a life cycle data item called out by DO-178B, the SCM archives are also available as gzipped ’tar’ files of ’cvs’ repositories. These archives contain not only all source code and documentation for the software release, but also the historical antecedents. In other words, they contained all of the source code and documentation for prior releases as well.

In summary, all life cycle data items are available in an on-line downloadable format rather than as hardcopies.

5.1.2.4. 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. Schedule

Because libBirds is a library reusable by developers of airborne software, rather than for use in any specific standalone hardware-based device, there is no obvious reason for any scheduled interactions with certification authorities.

The libBirds library is not a commercial project with a planned release date or other milestones. Hence it can be released whenever it happens to be ready.

In summary, there is no relevant scheduling data that can be presented for the initial development effort.

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 qualifiation, 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 tools 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.

8. GNU Free Documentation License

Version 1.1, March 2000

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. Secondarily, 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.

9. GNU Lesser General Public License

Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user’s freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library. b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change. c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License. d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above

provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.) b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with. c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution. d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place. e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above. b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library 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.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public 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.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME

THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS