University of Dayton Law Review

Volume 34 | Number 2 Article 4

1-1-2009

Toward Open Source Hardware

John R. Ackermann
University of Michigan

Toward Open Source Hardware

Cover Page Footnote

The author would like to especially thank Kirk Johnsen and Robert Lech for their assistance and support throughout the lengthy process that led to the Open Hardware License, and to this article.

TOWARD OPEN SOURCE HARDWARE

John R. Ackermann[*]

I. INTRODUCTION

Open Source Software, that is, software distributed and licensed in a way that allows users to modify and further distribute its source code,[1] has become an accepted part of the computer world. Most people involved in computer law today are familiar with the community-oriented philosophy behind Open Source Software licenses such as the Free Software Foundation's General Public License.[2] The purpose of such licenses is to ensure that software users retain the freedoms described in the Free Software definition,[3] in particular, they seek to ensure that Free Software cannot be "taken proprietary" by its incorporation into programs whose source code is not made available. At its most fundamental, the goal of licenses like the GPL is to foster a community where those who benefit from the work of others in turn contribute their improvements to that community. A similar movement, inspired by many of the same concerns that drove those software developers, has taken shape among people involved in electronic hardware design efforts on a collaborative basis: the idea of Open Source Hardware.

---

[*] The author received his J.D. from the University of Michigan Law School. He would like to especially thank Kirk Johnsen and Robert Lech for their assistance and support throughout the lengthy process that led to the Open Hardware License, and to this article.

[1] The Open Source Initiative lists ten characteristics of Open Source software including free redistribution, availability of source code, and the right to create modifications and derivative works. Ken Coar, *Open Source Initiative, The Open Source Definition (Annotated), Version 1.9*, http://www.opensource.org/docs/definition.php (July 24, 2006). "Free Software" is a somewhat more restrictive term which obligates Free Software users to follow certain rules concerning the modification and distribution of the software. *See infra* n. 3. A general term that informally encompasses both types of software is Free and Open Source Software ("FOSS"). *See generally* Wikipedia, *The Free Ency., Free and Open Source Software*, http://en.wikipedia.org/wiki/Free_and_open_source_software (last modified March 19, 2009). Free Software is a subset of Open Source Software.

[2] Free Software Foundation, *GNU General Public License, Version 2*, http://www.gnu.org/licenses/old-licenses/gpl-2.0.html (last updated Jan. 30, 2009).

[3] The Free Software Definition defines four "freedoms" which characterize Free Software:

> The freedom to run the program, for any purpose (freedom 0). The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this. The freedom to redistribute copies so you can help your neighbor (freedom 2). The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Free Software Foundation, *GNU Operating System, The Free Software Definition*, http://www.gnu.org/philosophy/free-sw.html (last updated Apr. 26, 2009).

Many people believe that the "build it yourself" days of electronics are over because of both the extreme complexity and the extreme miniaturization of modern technology. Some of us have a mental picture of old-time radio hobbyists, often ham radio operators,[4] hunched over a smoking soldering iron in a basement workshop, building a radio from plans in a magazine -- Open Source Hardware in an earlier form – and assume that those days are gone forever.

Two facts undercut this assumption. First, software programs now available at affordable prices or for free (in fact, some are GPL licensed[5]) allow the design of complex circuits and printed circuit board layouts.[6] Second, an industry has grown up around low-volume production of printed circuit boards. One can email the output files created by one of those design programs to a company which will use the files to make printed circuit boards in a short time for a low cost. For example, one company will make three customized circuit boards for approximately fifty dollars and ship them within a day or two of receiving a customer's data files.[7]

As a result, a number of individuals and groups now create sophisticated electronic designs for their own use, and, driven by some of the same motivations that drive Open Source Software developers,[8] often want to make these designs available to others. One of the same concerns that drove the creation of the Open Source Software community is now plaguing these hardware developers. They want to make their designs available to the public, but they do not want those designs co-opted by parties unwilling to contribute back to the community. A group of these developers in the ham radio community asked the author of this article to create an Open Source Hardware license that implements a Free and Open Source philosophy in the same way that the GNU General Public License does for software. This effort led to the present article that will first

---

[4] Ham radio, more formally known as "amateur radio" is a hobby in which participants operate, and often build their own radio transmitters and associated equipment. Radio amateurs are licensed by the communications regulators in their country. The hobby began over a century ago, and amateur radio operators were associated with many of the technical advances of the radio art during the 20th and 21st centuries. *See generally* The American Radio Relay League, *Welcome to the World of Ham Radio*, http://www.wedothat-radio.org (accessed May 17, 2009); The American Radio Relay League, *Science, Technology, Experimentation...We Do That!...with Amateur Radio*, http://www.wedothat-radio.org/wedothat/ (accessed May 17, 2009).

[5] *See e.g.* Seul Project, *GEDA Project, What is GEDA?*, http://gpleda.org/ (accessed May 15, 2009).

[6] Printed circuit boards, which provide a mounting surface and conductive traces to interconnect components, have almost entirely replaced traditional "point to point" wiring of parts mounted in a metal chassis. *See generally* Wikipedia, *The Free Ency., Printed Circuit Board*, http://en.wikipedia.org/wiki/Printed_circuit_board (last modified Feb. 11, 2009).

[7] ExpressPCB, *Manufacturing Specs*, http://www.expresspcb.com/ExpressPCBHtm/Specs.htm (accessed Feb. 20, 2009). Note that the price listed is for manufacturing the boards only, and does not include the components that mount on the the boards.

[8] In a 2007 article, Linus Torvalds, creator of the Linux software, noted the attractiveness of Open Source in the context of scientific inquiry. Bruce Byfield, *Linux Today, Linus Explains Why Open Source Works*, http://www.linuxtoday.com/developer/2007081301726INKN (Aug. 13, 2007). That context seems particularly applicable to Open Source Hardware developers.

describe the process of developing an electronic product based on a printed circuit board and describe how that process differs from software development. Second, it will describe the legal implications of the hardware development process. Finally, this article will describe an attempt to reconcile the legal and practical implications of applying an Open Source philosophy to hardware development, which culminated in the TAPR[9] Open Hardware License[10] and the Noncommercial Hardware License.[11]

II. THE HARDWARE DEVELOPMENT PROCESS

To understand why an Open Source Hardware license requires a different legal framework than that used for Open Source Software, it is necessary to understand the differences in the way hardware and software are created.

Leaving aside high level architectural and design issues, software development follows a relatively straight-forward path. The programmer[12] writes source code into files that are stored on the computer in a human-readable format. Following good design practice, the programmer will write code in a modular fashion, resulting in multiple, perhaps even hundreds or thousands of, individual source code files.[13]

When it is time to convert human-readable software source code

---

[9] Tucson Amateur Packet Radio Corporation, which brands itself using the acronym TAPR, is nonprofit corporation with an international membership base. Founded in the early 1980s, TAPR has evolved into a research and development organization supporting digital and computer-based technology in amateur radio. See generally TAPR, *Organization*, http://www.tapr.org/organization.html (last updated May 26, 2007).

[10] The TAPR Open Hardware License is included for reference in appendix 1 to this article. *See also* TAPR, *Publications: Open Hardware License, The TAPR Open Hardware License*, http://tapr.org/ohl.html (last updated May 26, 2007); TAPR, *Publications: Noncommercial Hardware License, The TAPR Noncommercial hardware License*, http://www.tapr.org/NCL (last updated May 26, 2007).

[11] An important note about subject matter, this article will *not* address the licensing of code for programmable logic devices such as Field Programmable Gate Arrays ("FPGAs") and Complex Programmable Logic Devices ("CPLDs"). These devices, programmable using languages such as VHDL and Verilog, are key components of modern electronic design. However, the code that is loaded into them is, from a legal perspective, much like any other software. An open source community has developed around code developed for these devices, using traditional licenses such as the GPL. *See e.g.* Open Cores, *Frequently Asked Questions*, http://www.opencores.org/?do=faq (accessed May 15, 2009). This paper is about the physical implementation of electronic hardware, not the code that may be loaded onto that hardware.

[12] Software development is often a team effort and there may be many programmers contributing code; the kernel of the GNU/Linux operating system has more than one thousand individuals contributing to every release. The Linux Foundation, *Linux Kernel Development*, http://www.linuxfoundation.org/publications/linuxkerneldevelopment.php (Apr. 2008). For simplicity, this article will refer to the singular programmer who represents this team. This, by the way, is another difference between the open software and open hardware communities. While a hardware project can certainly have more than one contributor, the tools commonly available to experimenters are not conducive to merging the input of multiple developers; thus, one person tends to control each phase of the project.

[13] For example, Linux kernel version 2.6.27 includes over 24,000 separate files containing about 8.75 million lines. Michael Florian Schönitzer, *Linux Kernel Statistics*, http://www.schoenitzer.de/lks/lks_en.html (accessed May 15, 2009).

into computer-understandable binary code,[14] a master configuration file (often called a "makefile") instructs software tools, such as a compiler[15] to process the source code files and create the binary result. Assuming all the necessary source code files are syntactically correct and the necessary supporting files are available, the result will be one or more programs ready for use. The "build" process may include many discrete steps, but in most cases, the makefile triggers and manages these discrete steps without any user intervention. For even a complex task such as compiling the Linux kernel (the core process that runs a Linux-based computer[16]) from its millions of lines of source code, the compilation task is largely finished after typing "make" at the console. The programmer need only revisit the computer some seconds, minutes, or hours later to determine if the source code compiled without errors, and to install or package the program.

In contrast, creation of electronic hardware[17] requires discrete and potentially independent steps. The development process involves tasks that can only partially be turned over to Electronic Design Automation ("EDA")[18] software; these tasks require a number of skills and potentially a number of different tools. The nature of these steps and tools affect the existence of intellectual property rights during the creation of the electronic hardware.

The first step in the design process is the creation of a schematic diagram (Figure 1).[19] The schematic is a graphical representation of the components in the circuit and the wires connecting them. It can be imperfectly analogized as falling between a flow-chart depiction and the actual source code of a computer program. The schematic uses a standardized set of symbols[20] to represent the components of the circuit, such as resistors, capacitors, transistors, and integrated circuits. The schematic assigns unique identifiers to each, and where relevant, shows the value of the component (e.g., a 10-kilohm resistor).

---

[14] *See generally* The Linux Information Project, *Source Code Definition*, http://www.linfo.org/source_code.html (last updated Feb. 14, 2006) (providing definitions of source and object code).

[15] A compiler is a program which does the actual conversion from source to binary code. *Id.*

[16] The Linux Information Project, *Kernel Definition*, http://www.linfo.org/kernel.html (last updated May 31, 2005).

[17] This article focuses on the creation of electronic products. Open Source Hardware design principles apply to other kinds of products, but the development process may be quite different.

[18] Electronic Design Automation is a general term used for software that performs tasks such as schematic capture and circuit board layout. *See generally* Wikipedia, *The Free Ency., Electronic Design Automation*, http://en.wikipedia.org/wiki/Electronic_design_automation (last modified Mar. 28, 2009).

[19] All figures appear at the end of this article. In order to retain legibility in printing, the figures in this article show a simple product, one designed by the author. TAPR, *Kits: Clock-Block Clock Synthesizer, TAPR Clock-Block*, http://www.tapr.org/kits_clock-block.html (last updated Nov. 12, 2008). Many amateur hardware development projects are far more complex.

[20] For example, IEC standard 60617 and ANSI standard Y32 are standard symbols set in wide use. ARRL The National Association for Amateur Radio, *Help Desk, Schematic Symbols Used in ARRL Circuit Diagrams*, http://www.arrl.org/qst/qs4hd.pdf (accessed May 15, 2009).

However, a schematic diagram provides virtually no information about the physical arrangement and interconnection of the parts; it is a purely logical depiction of the circuit. More is needed to turn modern, complex circuits into reproducible products.

Printed circuit boards are the almost universal basis for electronic manufacturing today (Figure 2).[21] Printed circuit boards have almost completely replaced the older point-to-point wiring used prior to the 1960s. There are several advantages of using printed circuit boards over point-to-point wiring. First, printed circuit boards allow for greater automation in manufacturing. Point-to-point wiring was a manual, labor-intensive process. In contrast, printed circuit boards eliminate the wiring process; the electrical connections are part of the board. The board also provides the mechanical structure to hold the components of the circuit, and modern manufacturing techniques can automatically place and solder[22] those parts. The second great advantage of the printed circuit board is that it can allow for much greater density than older wiring methods, which means that more components can fit in a smaller space. Printed circuit board techniques make possible the use of integrated circuit chips that may be barely an inch square yet contain millions of transistors and have hundreds of signal connections.[23] Accompanying components can be as small as two one-hundredths by one one-hundredth of an inch[24] –literally microscopic in size. Those chips and components are at the heart of modern products such as mobile phones, personal computers, and countless other devices.

Printed circuit boards are made of a base material that will not conduct electricity, such as epoxy or fiberglass, with a thin layer of copper deposited on one or both sides. Boards with more than two copper layers are made by sandwiching layers of insulator and copper together. Through a photochemical process similar to the one used to make plates for printing presses (hence the "printed" part of the name), the copper is etched away leaving behind tracks (or "traces") that are the equivalent of the wires shown on a schematic diagram. Holes drilled at the appropriate places accommodate the wire leads attached to some components, while other components mount on the surface of the board without using wire leads.

---

[21] This photo is of a completed unit of the schematic shown in Figure 1.

[22] Soldering is the technique of using a metal alloy that is electrically conductive and has a low melting point to join wires and electronic components together. *See generally* Electronix Express, *Better Soldering*, http://www.elexp.com/t_solder.htm (last modified Jan. 4, 2002).

[23] One of the chips used in the HPSDR project described below, the Altera Cyclone II FPGA, has 208 pins and is 30.6 millimeters – just over one inch – square. Altera Corp., *Literature: Cyclone II Devices*, http://www.altera.com/ literature/lit-cyc2.jsp (accessed May 17, 2009).

[24] That is the size of "0201" surface mount packages available for simple components such as resistors and capacitors. Most designs use sizes somewhat larger than that, but packages about one tenth inch in size are considered largely obsolete. Adrio Communs. Ltd., *Radio Electronics.com, SMT Component Packages*, http://www.radio-electronics.com/info/data/smt/smt_packages.php (accessed May 15, 2009).

Components are soldered to the traces either by hand or by machine. [25] The solder provides both electrical connectivity and mechanical support to hold the components in place.

The software and hardware design processes diverge when the logical design expressed by the schematic diagram turns into the physical embodiment of a circuit board. If going from schematic to printed circuit board is equivalent to compiling source into object code, the process for turning the schematic diagram into the physical circuit board is significantly less automated and requires significantly more manual intervention than the equivalent process for software. There are several important reasons for these differences.

First, while the schematic diagram is a complete representation of the electrical structure of the circuit, there is much information it does not contain. For example, it does not describe the physical location of components on the circuit board, nor does it show the physical characteristics of the individual components.[26] Proper performance, particularly of high frequency radio or high speed digital circuits, requires careful attention to the physical placement of components and their interconnection. External factors, such as size constraints and connection requirements, affect the arrangement of the components. The printed circuit board designer must understand these issues in order to arrange the placement of components and traces in a way that addresses these externally imposed requirements, while still adhering to the schematic diagram's map of electrical connections.

Second, there is a degree of artistic, or at least creative, skill involved in laying out a printed circuit board. One fundamental challenge is that the copper traces that connect components together cannot touch each other except where the schematic calls for that. The traces must be "routed" in a way that avoids unwanted contact.[27]

Before the advent of computer-based design tools, the "mask" from which one etched a printed circuit board was created by laying black tape on a sheet of clear acetate. The work involved not only the skills described above, but a fair degree of artistry and talent with a sharp knife. Today,

---

[25] Specialized machines, both because of their speed and their ability to handle microscopic components, almost invariably assemble commercial electronic products. However, it is still possible and practical in low-volume applications to install most of these parts by hand if the proper techniques are used. *See e.g.* SparkFun Electronics, *SMD How To - 1*, http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=36&page=1 (Sept. 2, 2006). The author's experience is that it is more difficult to get over the fear of working with these tiny parts than to actually solder them.

[26] However, an integrated EDA package that offers both schematic drawing and circuit board layout functions may define and store the necessary information for later hand-off to the layout module.

[27] An easy way to simplify layout is to add more layers to the board and thus more surfaces on which signal traces may be routed. There is a cost to this, however: using standard techniques, going from two to four layers about doubles the cost of each board. PCB Design.org, *Understanding PCB Layers*, http://www.pcbdesign.org/pcb-layout/understanding-pcb-layers/ (accessed May 15, 2009).

computer programs have done away with literally "taping up" a board, and dramatically changed the process of translating a schematic diagram into a circuit board. The automated circuit board design process includes two broad steps: first, the designer creates a logical depiction of the components and their interconnection using a "schematic capture" program, and then he translates that drawing into a physical design for the circuit board using a "layout" program. Some developers use separate programs for the schematic capture and board layout phases, while others use integrated EDA tools that handle the entire process.

The Modern Design Process

The next few paragraphs describe the flow of the modern electronic product design process and the outputs that each step in that process creates. These outputs form the basis for the intellectual property analysis that follows.

The schematic capture program works something like the flowchart design programs, such as Microsoft Visio,[28] used in business settings. The designer selects component symbols from a library, places them on the workspace, and draws lines representing the wires that will connect the components together. The software can generate several output files. It will create a file, usually in a proprietary format, defining the entire schematic drawing and ancillary information. In addition, it will usually have the ability to export the schematic diagram in a common graphic format such as portable document format ("PDF").

However, the most important aspect of the schematic capture program output is a "netlist." A netlist is a file that describes the electrical connectivity of the design by defining each set of connections (or "net") contained in the schematic. For example, Figure 3 shows a few lines from a simplified netlist file. These lines indicate the interconnection of components C8, IC1, R1, and R8, into one net, as well as components L1 and R21 into another.[29] In other words, each net signifies a group of component connections that are electrically tied together. The netlist file may also contain information describing the electrical value and physical type of the components contained in the schematic; alternatively, this information may be exported in a separate file.[30] The netlist and component

[28] Microsoft Corp., *Microsoft Office Online, Microsoft Office 2007 Visio Product Overview*, http://office.microsoft.com/en-us/visio/HA101656401033.aspx (accessed May 15, 2009).

[29] By convention, each component on a schematic diagram is given a unique identifier consisting of one or more letters that indicate the type of component, and a number that indicates its unique sequence among the other components of its type. In this example, "C" signifies a capacitor, "R" a resistor, "IC" an integrated circuit, and "L" an inductor.

[30] In some cases, where the schematic capture and circuit board layout software are completely independent, all component information may be entered during the layout process, with none coming from the schematic capture program.

information files are inputs to the printed circuit board layout program.

The layout software uses an additional set of inputs: component libraries that provide detailed information about the size, the shape, and the connection pins for each physical type (or "variant") of component. Information from these libraries merges into the final design. For example, the component file generated by the schematic program may call for a type 2N3904 transistor in a type SOT-23 package. A component library will contain the detailed physical characteristics, such as its dimensions and where its mounting leads are located, for all the available variants of that transistor. Based on the component variant the designer selects, the software will import the necessary information from the component libraries to allow placing that component on the board.

Much like the libraries used in software programming, component libraries may come from a number of sources. EDA software suppliers provide component libraries as part of their products, users create their own libraries as necessary to deal with new components, and user communities share these libraries. The libraries are usually specific to a particular EDA program, though in some cases libraries for one program can be converted for use with another.

Circuit board layout is an interactive, graphically oriented process. The layout software uses the netlist and component information files as its initial inputs. The designer uses the computer keyboard and mouse to move components around the representation of the board appearing on his computer screen, and to place the traces that interconnect those components. Board layout is often like a three-dimensional game as the designer juggles placement not only of the component, but also of the traces that may be stacked one above the other on as many as forty layers.[31]

To create the circuit board using EDA software, the designer first places the components, shown as an outline of their relative size and shape, onto a workspace that represents the surface of the board. The designer then routes the copper traces, using all the available layers, to interconnect them. Modern design tools often have an "autoroute" feature that will attempt to automatically place the traces, but even the best of these have limitations. Usually the autoroute output, if used at all, serves as a starting point for the designer, rather than as the finished product. Figure 4 shows a graphic view of a circuit board layout, including outlines of the components (which are printed on the board to aid assembly, but do not have any electrical function), holes and pads for component mounting, the traces placed on the copper layers and other information silkscreened on the board to aid in

[31] *E.g.* San Francisco Circuits, Inc., *We Save You Time and Money*, http://www.sfcircuits.com/ (accessed May 15, 2009). Forty layers are quite extreme, but such boards are commercially available. Boards with two or four layers are much more common.

assembly or use.[32]

The board layout program, like the schematic capture software, ordinarily saves its work in a file with a proprietary format. It can also output various images of the board layout in one or more common graphic formats. Its primary output, however, is a set of "Gerber" files in the industry standard format used by printed circuit board manufacturers.[33] One or more Gerber files describe each aspect of the physical implementation of the board. For example, one file instructs the computer-aided manufacturing machine to etch the copper traces on the top side of the board. Another file describes the traces on the bottom of the board. A third describes the location and size of each hole to be drilled for component mounting.

Given a complete set of Gerber files, a manufacturer can create a printed circuit board, or, for that matter, 100,000 of them. Because these files are in a standard format and easy to transmit electronically, a new supply chain has developed based on rapid and inexpensive production. A number of companies allow the entire order process to take place over the internet. A customer may receive a package of boards in the mail shortly after uploading the Gerber files. This capability, coupled with the availability of free or inexpensive schematic capture and board layout software, has largely driven the recent boom in non-commercial hardware development.

III. INTELLECTUAL PROPERTY AND HARDWARE DEVELOPMENT

Copyright suffuses software and its development. Courts in the early days of computers struggled to fit the new technology of software into a copyright paradigm, questioning in particular whether binary code that only a computer could comprehend was a form of expression subject to copyright protection, or instead was an idea or process protectable, if at all, only under the patent act.[34] The Third Circuit's decision in *Apple Computer, Inc. v. Franklin Computer Corp.*[35] in 1983 effectively resolved that

---

[32] This is the layout of the schematic diagram shown in Figure 1.

[33] AP Circuits, *Gerber Data Format*, http://www.apcircuits.com/resources/information/gerber_data.html (accessed May 15, 2009).

[34] *See e.g. Apple Computer, Inc. v. Franklin Computer Corp.*, 545 F.Supp 812, 823 (E.D. Penn. 1982) (refusing to grant a preliminary injunction against the copying of Apple operating system software embodied in floppy disks and Read Only Memory chips in light of defendant Franklin's arguments that the binary code was not a copy of an "original work" or that a ROM chip was a "machine" incapable of copyright protection).

[35] 714 F.2d 1240, 1254 (3d Cir. 1983). After noting of the Court's holding below that "[i]t is difficult to discern precisely why the district court questioned the copyrightability of the programs at issue," the Circuit Court found that the District Court presented three questions relevant copyrightability: "(1) whether copyright can exist in a computer program expressed in object code, (2) whether copyright can exist in a computer program embedded on a ROM, [and] (3) whether copyright can exist in an operating system program . . . ." *Id.* at 1246. In its decision, the court answered all these questions in the affirmative. *See generally id.*

argument by unequivocally holding that computer programs, whether in source or binary form, and whether viewed as application or operating system software, are literary works subject to the copyright act.

Because copyright clearly governs software, most drafters build software licenses, whether for Open Source or proprietary programs, primarily around copyright concepts. Some software licenses, such as the Free Software Foundation's General Public License ("GPL"), expressly claim to be purely copyright licenses, and disclaim any contractual nature.[36] For example, the GPL is based on the premise that "[l]icenses are not contracts: the work's user is obliged to remain within the bounds of the license not because she voluntarily promised, but because she doesn't have any right to act at all except as the license permits."[37] Others, especially in commercial contexts, include significant contractual elements extending beyond mere control of the exclusive rights granted by the copyright act. But in virtually all cases, they rely on copyright to establish the author's underlying right to control the use and dissemination of the program.

Protection of Open Source Hardware

The Open Source Hardware developer must first consider just what he seeks to protect. Broadly, the outputs of a hardware project divide into two categories: (1) the documentation that reflects the design, and from which one may manufacture the product; and (2) the products that are in fact manufactured from that documentation. In some ways these categories map to source code and binary computer code, but unlike software, where the same copyright analysis applies to both those manifestations of the program, there may be quite different legal issues at play when considering documentation versus product protection, and even among various elements of the documentation.

There are as many opinions about the goals of an Open Source Hardware license as there are hardware designers, but the following discussion assumes that the designer seeks to tie both the documentation, and the product that results from it, to a set of goals that are, as much as feasible, consistent with the four freedoms defined by the Free Software Foundation's Free Software Definition.[38] This means that the designer wants to ensure at a minimum that those who distribute the documentation for an Open Source Hardware project, and products based on that documentation, comply with two obligations: (1) that they make the documentation which they received under the Open Source Hardware license available to all; and (2) that they likewise make their modifications

[36] *GNU General Public License*, *supra* n. 2.

[37] Eben Moglen, *GNU Operating System, Enforcing the GNU GPL*, http://www.gnu.org/philosophy/enforcing-gpl.html (Sept. 10, 2001).

[38] *The Free Software Definition*, *supra* n. 3.

to that documentation available to all on the same terms as the original work.[39] These goals track those of the GPL, which requires that those who distribute GPL'd software: (1) make the source code available to all under the GPL; and (2) license their modifications under the GPL as well, thereby ensuring that those modifications are also available to all. It is important to note that neither set of requirements include restrictions on modification or use that is unaccompanied by distribution.[40]

The GPL purports to achieve these goals solely through the mechanism of a copyright license. It grants a limited license to exercise the rights reserved to the copyright holder by statute. These rights include the right to copy, the right to distribute, and the right to create derivative works (e.g., modifications and executable versions of source code).[41] Can this model be applied to Open Source Hardware?

Protecting Hardware Designs and Implementations

Underlying all the discussion the follows is the crucial concept that copyright does not protect ideas, but only the expression of ideas.[42] "Unlike a patent, a copyright gives no exclusive right to the art disclosed; protection

---

[39] As discussed later in this article, the real economic cost of tangible hardware products means that forcing free access to such hardware is not a rational goal of Open Source Hardware.

[40] GPL Version 2 states that:

> Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted. . . .
>
> . . .
>
> [and][y]ou must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

Free Software Foundation, *GNU Operating System, GNU General Public License, Version 2.0* § 0, § 2(b), http://www.gnu.org/licenses/old-licenses/gpl-2.0.html (updated Jan. 30, 2009) (emphasis added).

[41] These are among the rights granted by the copyright act, under which a copyright holder has the exclusive right to the following:

> (1) to reproduce the copyrighted work in copies or phonorecords; (2) to prepare derivative works based upon the copyrighted work; (3) to distribute copies or phonorecords of the copyrighted work to the public by sale or other transfer of ownership, or by rental, lease, or lending; (4) in the case of literary, musical, dramatic, and choreographic works, pantomimes, and motion pictures and other audiovisual works, to perform the copyrighted work publicly; (5) in the case of literary, musical, dramatic, and choreographic works, pantomimes, and pictorial, graphic, or sculptural works, including the individual images of a motion picture or other audiovisual work, to display the copyrighted work publicly; and (6) in the case of sound recordings, to perform the copyrighted work publicly by means of a digital audio transmission.

17 U.S.C. §106 (2006).

[42] 17 U.S.C. §102(b) (2006); *see generally Baker v. Selden*, 101 U.S. 99 (1879); *Mazer v. Stein*, 347 U.S. 201 (1954).

is given only to the expression of the idea—not the idea itself."[43] There is nothing in the copyright law to stop one from implementing the circuit – in other words, the idea – described by a schematic diagram, even if that diagram is clearly subject to copyright.[44] This is the fundamental difficulty in creating a license for Open Source Hardware: it is the idea, or at least its physical implementation, that we wish to protect, and copyright is clearly not a universal tool to accomplish this.

The first approach likely to occur to one who wishes to create an Open Source Hardware license, therefore, is to use patent law, rather than copyright law, as the framework for protection. One can patent inventions contained in the design (assuming they meet the required thresholds of novelty,[45] non-obviousness,[46] and usefulness[47]), and allow manufacturing and distribution of products practicing those inventions only subject to the terms of a patent license. With such protection in place, the question of whether copyright can protect the product's design files is reduced to secondary importance. This theory is not faulty, but it has three practical limitations.

First, the cost required to obtain a patent is simply beyond the means of most individuals. Including legal fees, obtaining a simple U.S. patent can cost $4,000 to $6,000.[48] Complex patents can cost $15,000 or more, assuming no complications.[49] Once the patent issues, ongoing maintenance costs of at least $3,000 add to the expense.[50] Even large corporations must make tactical decisions, taking into account the cost of each patent application, its prosecution, and its maintenance, as to which invention disclosures to pursue. For a single developer working on his or

---

[43] *Mazer*, 347 U.S. at 217 (citing *F. W. Woolworth Co. v. Contemporary Arts*, 193 F.2d 162, 164 (1st Cir. 1952)).

[44] One could argue that the design files used to create a printed circuit board are analogous to blueprints used to construct a building. However, architectural works – both the blueprints and the building – have a unique status under the copyright act. The Architectural Works Protection Act of 1990, Title 7 of PL 101-650, amended the copyright act to provide specific protection for both architectural drawings and the buildings constructed from them. An "architectural work" is defined as "the design of a building as embodied in any tangible medium of expression, including a building, architectural plans, or drawings. The work includes the overall form, as well as the arrangement and composition of spaces and elements in the design, but does not include individual standard features." 17 U.S.C. §101. Such works are entitled to copyright protection as "pictorial, graphic, and sculptural works." *Id.* The Copyright Office has defined "buildings" for copyright purposes as "humanly habitable structures that are intended to be both permanent and stationary, such as houses and office buildings, and other permanent and stationary structures designed for human occupancy, including but not limited to churches, museums, gazebos, and garden pavilions." 37 C.F.R. §202.11(b) (2005). Whether for good or ill, "architectural works" and "buildings" as so defined cannot encompass printed circuit boards or the design files used to manufacture them.

[45] 35 U.S.C. §102 (2006).

[46] 35 U.S.C. §103.

[47] 35 U.S.C. §101.

[48] IP Watchdog, Inc., *Cost of Obtaining a Patent*, http://www.ipwatchdog.com/patent/patent-cost/ (accessed May 15, 2009); BasicPatents.com, *What Does It Cost to Obtain a Patent?*, http://www.basicpatents.com/patcost.htm (accessed May 15, 2009).

[49] *Id.*

[50] *Id.*

her own time, or even a group of them working jointly, those costs are simply prohibitive – especially when one considers the kitchen table discussion likely to occur over the idea of spending thousands of dollars to implement a license for a "free" hardware design!

Second, patents may also be impractical because of the time it takes to obtain them. According to the U.S. Patent and Trademark Office, the average time (as of 2009) to obtain a patent is over twenty-four months.[51] A non-commercial design may go from inception to distribution (and perhaps to obsolescence) within a few months, and receiving a patent grant long after the developer has moved on to other things may be gratifying but not highly relevant.

Finally, for better or worse, there is a strong disdain for patents among much of the non-traditional hardware development community, and some designers simply may not accept a license based on the threat of patent enforcement. However, as shown below, patent rights may play an unorthodox role in implementing an Open Source Hardware license.

Given the lack of patent protection as a practical framework, let us consider copyright. As discussed in more detail below, the copyright analysis for Open Source Hardware documentation is relatively straightforward. However, we must also consider whether the printed circuit board that results from that documentation is itself subject to copyright protection.

The copyright act makes clear that copyright in a design cannot normally control the physical realization of that design, because as noted above copyright does not extend to the implementation of ideas expressed in a copyrighted work.[52] However, a printed circuit board – the primary output of the design process – does not appear greatly dissimilar to the "pictorial, graphic and sculptural works"[53] for which copyright is available. So perhaps copyright can be a useful tool to protect the printed circuit board, and thereby establish the basis for an Open Source Hardware licensing model. The following pages discuss the possibilities and pitfalls of such an approach.

Copyright Principles Applicable to Open Source Hardware

Two distinct but closely related concepts control the question of

---

[51] U.S. Patent and Trademark Office, *How Long Does It Take for a Patent Application to Be Processed*, http://www.uspto.gov/main/faq/ (Aug. 14, 2003).

[52] "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. §102(b).

[53] "Pictorial, graphic, and sculptural works" include two-dimensional and three-dimensional works of fine, graphic, and applied art, photographs, prints and art reproductions, maps, globes, charts, diagrams, models, and technical drawings, including architectural plans." 17 U.S.C. § 101.

whether Open Source Hardware documentation, or the printed circuit boards that result from use of that documentation, are subject to copyright. Both arise from the fundamental and familiar limitation on the scope of copyright described above.[54]

Useful Articles

First, useful and utilitarian articles, and functional elements, are excluded from copyright protection.[55] The Copyright Act defines a useful article as one having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information.[56] The court in *Incredible Technologies, Inc. v. Virtual Technologies, Inc.* stated that:

> The exclusion of functional features from copyright protection grows out of the tension between copyright and patent laws. Functional features are generally within the domain of the patent laws. . . . [A]n item may be entirely original, but if the novel elements are functional, the item cannot be copyrighted: although it might be eligible for patent protection....[57]

Idea Expression Dichotomy

Second, the idea-expression dichotomy, which is the "most fundamental axiom of copyright law," denies copyright protection where the expression and the idea it expresses are inseparable, even if the expression is not entirely functional.[58] This concept was first articulated in *Baker v. Selden*, holding that "where the art [that the book] teaches cannot be used without employing the methods and diagrams used to illustrate the book, or such as are similar to them, such methods and diagrams are to be considered as necessary incidents to the art, and given therewith to the public."[59] This concept is now embodied within the copyright act, which states that "[i]n no case does copyright protection for an original work of authorship extend to

---

[54] *CCC Info. Servs., Inc. v. Maclean Hunter Mkt. Reps., Inc.*, 44 F.3d 61, 68 (2d Cir. 1994).

[55] The Copyright Act definition of a pictorial, graphic, or sculptural work states that:

> Such works shall include works of artistic craftsmanship insofar as their form but not their mechanical or utilitarian aspects are concerned; the design of a useful article, as defined in this section, shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article.

17 U.S.C. § 101; *see also ADA v. Delta Dental Plans Assn.*, 126 F.3d 977, 980 (7th Cir. 1997).

[56] 17 U.S.C. § 101.

[57] 400 F.3d 1007, 1012 (7th Cir. 2005) (citing *Pivot Point Intl., Inc. v. Charlene Prods., Inc.*, 372 F.3d 913, 980 (7th Cir. 2004)) (emphasis omitted).

[58] *Feist Publications, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 344 (1991).

[59] 101 U.S. at 103.

any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work."[60]

The idea-expression dichotomy is often analyzed through two subsidiary doctrines, each intended to prevent copyright in a work from creating a monopoly in the idea expressed by that work. Together they help determine the extent to which a functional object, such as an electronic design as implemented in a printed circuit board, that may also contain expressive elements, may be protected by copyright.

Merger Doctrine

The merger doctrine directly addresses the idea-expression dichotomy by denying copyright to a work if the idea underlying that work can be expressed in only one way. In *CDN Inc. v. Kapes*, the court stated that:

> [P]rice is an idea of the value of the product, which can be expressed only using a number. Thus the idea and the expression merge and neither qualifies for copyright protection. This is the doctrine of merger. The argument springs from a venerable principle of copyright law. Ideas, like facts, are not entitled to copyright. In order to protect the free exchange of ideas, courts have long held that when expression is essential to conveying the idea, expression will also be protected. 'When the "idea" and its "expression are thus inseparable, copying the "expression" will not be barred, since protecting the "expression" in such circumstances would confer a monopoly of the "idea" upon the copyright owner free of the conditions and limitations imposed by the patent law.'[61]

Put another way, "[t]o ensure free access to ideas, courts have applied the merger doctrine such that 'even expression is not protected in those instances where there is only one or so few ways of expressing an idea that protection of the expression would effectively accord protection to the idea itself.' "[62] By its nature, the merger doctrine applies primarily to literally copying; if there are multiple ways to express an idea such that non-literal copying is at issue, the merger doctrine is likely inapplicable.

---

[60] 17 U.S.C. 102(b).

[61] 197 F.3d 1256, 1261 (9th Cir. 1999); *Herbert Rosenthal Jewelry Corp. v. Kalpakian*, 446 F.2d 738, 742 (9th Cir. 1971); *see also Mazer*, 347 U.S. at 217; 17 U.S.C. § 102(b); *Baker*, 101 U.S. at 99.

[62] *N.Y. Mercantile Exch., Inc. v. Intercontinental Exch., Inc.*, 497 F.3d 109, 116-17 (2d Cir. 2007) (quoting *Kregos v. Associated Press*, 937 F.2d 700, 705 (2d Cir. 1991)).

Scènes à Faire

The *scènes à faire* doctrine casts a somewhat wider net. It does not measure a work's eligibility for copyright protection based on the literal test of "can I say this in another way?" but rather is a broader examination of the work in its context. Most commonly applied in literary settings, "[t]he *scènes à faire* doctrine in general excludes from copyright protection material that is 'standard,' 'stock,' or 'common' to a particular topic, or that 'necessarily follow[s] from a common theme for setting.' "[63] However, the doctrine is also applicable to computer software, where it operates to

> [E]xclude from protection against infringement those elements of a work that necessarily result from external factors inherent in the subject matter of the work. For computer-related applications, these external factors include hardware standards and mechanical specifications, software standards and compatibility requirements, computer manufacturer design standards, industry programming practices, and practices and demands of the industry being serviced.[64]

In other words, if the author of the work had only a single practical way to express the work's idea, that expression is not protectable. In particular, *scènes à faire* is relevant to software "header files"[65] and other programming elements where interoperability requirements may dictate the use of common data structures and naming conventions.

Creativity

A basic requirement for copyright protection is that the work contain at least some modicum of creativity, or conversely that it is sufficiently non-utilitarian, to meet the threshold for copyright protection as a literary work. The creativity necessary is not great because "[o]riginal, as the term is used in copyright, means only that the work was independently created by the author (as opposed to copied from other works), and that it possesses at least some minimal degree of creativity. . . . [T]he requisite

---

[63] *Autoskill Inc. v. Natl. Educ. Support Sys., Inc.*, 994 F.2d 1476, 1494 (10th Cir. 1993) (quoting Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* vol. 3, § 13.03[b][4], 13-70 (Matthew Binder 1992)).

[64] *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366, 1375 (10th Cir. 1997).

[65] "Header files are repositories of common definitions and declarations. If a programmer knows that an operating system contains a certain header file, then the programmer can simply refer to the header file and avoid reproducing all of the definitions and declarations." *Unix Sys. Laboratories, Inc. v. Berkeley Software Design, Inc.*, 1993 WL 414724 at *4 (D.N.J. Mar. 3, 1993).

level of creativity is extremely low; even a slight amount will suffice."[66]

The concepts of merger and *scènes à faire*, along with a consideration of whether the minimal creativity standards of the Copyright Act have been met, are the basis for determining the extent to which the outputs of the hardware design process may be subject to copyright protection. With this in mind, let us first consider protection of the documents (likely in the form of computer files) created in the process of designing an electrical circuit, and then the tangible realization of those documents in the printed circuit board itself.

The Schematic Capture Program

A schematic diagram (as shown in Figure 1) almost certainly is an original work containing enough creativity, or conversely is sufficiently non-utilitarian, to meet the threshold for copyright protection as a literary work.

The designer has significant choice in where to position the parts on the drawing sheet, in how to identify those parts, and in the various graphic elements used to make the diagram more understandable. There are however some constraints on the designer's creative urges. In particular, the use of standard symbol sets and component notation is virtually mandatory if she wishes comprehension of the schematic by anyone else.[67] That standardized notation enforces some limits on the designer's creativity, but does it nudge the work over the edge of the idea-expression cliff?

While the use of standard notation hints of *scènes à faire*, the wide discretion the designer retains in the arrangement of those symbols on the schematic page almost certainly results in a work of sufficient creativity to meet the requirement for at least a "thin" copyright.[68] Only one United States case appears to have ruled on the question of copyright in the schematic diagram of an electronic circuit.[69] The court in *Picker International Corp. v. Imaging Equipment Services, Inc.* held, without discussion or analysis, that a set of schematic drawings was subject to a valid copyright.[70]

In addition to the graphical output of the schematic diagram, the schematic capture program generates another primary output file -- the

[66] *Feist*, 499 U.S. at 345 (citing Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* vol. 1, §§ 201[A], [B] (Matthew Binder 1990)).

[67] *Schematic Symbols Used in ARRL Circuit Diagrams*, *supra* n. 21.

[68] A work is subject to a thin copyright if it contains relatively little creative content, and a work subject to a thin copyright is usually only protected against "virtually identical copying." *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1442 (9th Cir. 1994).

[69] A number of cases may be found which use the term "schematic" in relation to copyright, but apart from *Picker International* they all do so in the context of architectural drawings. *See e.g. Nelson-Salabes, Inc. v. Morningside Dev., LLC*, 284 F.3d 505 (4th Cir. 2002).

[70] 931 F. Supp. 18, 37 (D. Mass. 1995).

netlist -- which is actually more important than the graphical output in circuit board development; the diagram is useful to human beings, but the netlist is critical to the circuit board layout process. In contrast to the schematic diagram, the netlist file substantially lacks the element of design freedom.

As shown by Figure 4, the netlist is almost exclusively functional and contains little indication of creativity, much less human involvement. While the schematic representation of the circuit involves at least some creative judgment, the netlist strips away creativity. It simply contains the logical interconnections between components, devoid of any remnant of how the schematic diagram appears to the eye. Even if the netlist includes component information, that information is purely descriptive of the attributes of the component.

If one attempts to recreate the schematic diagram from a netlist and any associated component files, the outcome will almost certainly be a document that looks different than the original diagram; nonetheless, if skillfully performed, it will be a complete and accurate representation of the circuit described by the original. The idea (the logical interconnection of the circuit's components) and the expression of that idea (a list of those connections) have merged and the result cannot be protected under copyright.

A netlist file is perhaps analogous to a computer language header file, which contains functional elements such as interface information, but has only a minimal expressive component. The netlist may be a derivative work (in the form of a translation) of the schematic diagram, but that derivative is stripped of the creative content of the original. Like a header file, the netlist's idea and its expression have merged.[71]

While the schematic capture software exports its results in a standard graphic form as well as a textual netlist, it normally uses a unique (if not proprietary) format to store information for its own use. This native file normally contains all the information contained in the schematic diagram and has few external constraints on its structure. Therefore it should be subject to at least the same degree of copyright protection as the graphical form of the diagram.

The Circuit Board Layout Program

The design of the circuit board based on the netlist and other inputs requires a significant amount of skill and creative input -- arguably, more than is required for the development of the schematic layout because the arrangement of the schematic diagram is intended for human eyes only,

[71] *See generally Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992).

while the arrangement of the components and routing on the circuit board is, in fact, the arrangement of the physical product. The circuit board designer is constrained to lay out a board that implements the circuit described by the schematic, and the final result must be one that meets external constraints such as size, interconnection, and cost, and can be successfully manufactured. That is not always a simple task.

First, the designer must choose where to place the components on the surface of the board. External design criteria such as the size of the board, the location of input and output connections, and engineering requirements such as ensuring that components do not electrically interfere with one another, may dictate where the designer places the components.

Second, once the designer places the components, he must route the circuit wires (or "traces") to connect the components together. As discussed above, integrated circuit components may each have hundreds of individual pins (or connection points). Moreover, a single board may have several of these "dense" components, and thus there can be thousands of individual traces to be placed. Even with multiple layers available for routing determining how to arrange the traces to avoid trace conflicts is a complex and daunting task that requires skill and patience. As with the placement of components, the placement of traces involves both skill and creativity, and the finished result can resemble a work of abstract art.

Where external constraints significantly limit the designer's layout choices, the *scènes à faire* doctrine may be argued to limit the scope of copyright in the resulting work. *Scènes à faire* means that one may own the descriptive elements inherent in a genre: there are only so many ways for a computer game to depict a karate match, and through *scènes à faire* many games may share that depiction without infringing.[72] However, in the hardware design context the inherent elements are usually based upon electrical or physical interface requirements that are purely functional rather than descriptive, and as such are ineligible for copyright protection under any circumstances. For example, a connector with a specified size may need to be placed at a specified location on the board. While the designer may not claim copyright in such functional elements, neither should their inclusion as a portion of an otherwise creative design somehow render the whole to be functional.

A significant part of board layout comes down to a sense of design and efficiency, and in fact the greater the constraints imposed, the more skill and creativity may be required to work around them. The likelihood of two skilled designers independently producing identical boards based on the same schematic is negligible. The circuit board layout should be subject to

[72] *See generally Data East USA, Inc. v. Epyx, Inc.*, 862 F.2d 204 (9th Cir. 1988).

copyright, at least as strong as that which attaches to the schematic diagram.

As with the schematic capture program, the printed circuit board layout software generates several output files: a human-viewable graphical format that shows the placement of parts and traces (Figure 4); a machine-readable set of Gerber files that instruct specialized machines how to create the circuit board (Figure 5); and a unique native file format that is used by the program itself. Since the native file format includes all the creative content captured by the graphical output file, it should carry at least the same level of copyright protection as the graphical output file.

Gerber files are the inputs used to actually manufacture the printed circuit board and are thus key components in the design process. No published United States court decision appears to have addressed whether Gerber files are subject to copyright. Under the copyright principles discussed above, Gerber files, unlike the superficially similar netlist output from a schematic capture program, contain substantial protectable expression.

While the netlist contains only a logical description of connections, Gerber files contain a complete translation of the physical layout of the board. They include all the substantive information contained in the layout program's graphical output, and from them, one can recreate exactly the same layout as the original program. In fact, standalone Gerber viewer programs are available that will graphically show the copper layers, component outlines, and other features of a circuit board as they will appear on the manufactured board.[73] Perhaps more than any of the other file formats discussed in this article, Gerber files bear a resemblance to computer software source code. Like source code, Gerber files are the input to the process that produces the final desired output. Because they contain the same information as the graphical output of the design program, the Gerber files should likewise be considered literary works and be subject to the same degree of copyright protection as the graphical output files.[74]

In summary, a designer who wishes protect his documentation files should find that copyright laws provide at least some protection to the graphical outputs, native file formats, and Gerber file outputs of his design tools. However, the netlist file is likely to run afoul of the idea/expression dichotomy and be ineligible for copyright protection.

---

[73] *See generally* Micro Technology Services, Inc., *Free Gerber Viewer, Free Gerber CAM and PCB Viewers*, http://www.mitsi.com/pcb/free%20viewers.htm (last updated Oct. 25, 2008).

[74] It may be argued that Gerber files differ from source code in that they are almost always machine generated by layout programs rather than being directly generated by a human developer. However, automated programming techniques are used in software development. *See generally* Wikipedia, *The Free Ency., Automatic Programming*, http://en.wikipedia.org/wiki/Automatic_programming (last modified May 5, 2009). The way in which the code was generated should have no impact on its copyright status.

The Printed Circuit Board Itself

As discussed above, any copyright in the schematic diagram that describes an electronic circuit does not constrain someone from building that circuit; only patents can do that. The question of whether the printed circuit board itself is subject to copyright is an important one, if for no other reason than within the electronics industry there is a widespread assumption that it is. It is very common to see copyright notices on circuit boards.[75]

On first thought, it might appear that a circuit board should be subject to the same copyright scope as are the board design files that represent it, for after all the board is merely the design layers "printed" and stacked on the base material. The Copyright Act includes "pictorial, graphic, and sculptural works" within its scope,[76] and defines these as including "two-dimensional and three-dimensional works of fine, graphic, and applied art, photographs, prints and art reproductions, maps, globes, charts, diagrams, models, and technical drawings, including architectural plans."[77] However, to the extent such a work is mechanical or utilitarian in nature, it cannot be copyrighted because

> [Pictorial, graphic, and sculptural] works shall include works of artistic craftsmanship insofar as their form but not their mechanical or utilitarian aspects are concerned; the design of a useful article, as defined in this section, shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article.[78]

No reported United States case has determined whether a circuit board is so utilitarian that it is ineligible for copyright protection, and no scholarly journal appears to have considered the question.[79] In *Alcatel USA, Inc. v. DGI Technologies, Inc.*, the court upheld a jury finding of direct

---

[75] A note to those researching circuit board copyright cases: many of the video game infringement cases from the 1980s refer to copyright notices on the circuit board, or talk about the circuit board itself being infringing. However, those cases suffer from a lack of precision in language; their real subject matter was the audio-visual content of the game which resided in read-only memory chips mounted on the boards. The cases discuss infringement of that content, rather than of the board design. *See e.g. Red Baron-Franklin Park, Inc. v. Taito Corp.*, 883 F.2d 275 (4th Cir. 1989), *modified*, *Red Baron-Franklin Park, Inc. v. Taito Corp.*, 1989 U.S. App. LEXIS 19806 (4th Cir. Sept. 5, 1989). It is not clear from those cases whether the designers thought they were only protecting the ROM chips, or thought the notice protected the boards themselves.

[76] 17 U.S.C. § 102(a)(5).

[77] 17 U.S.C. § 101.

[78] *Id.*

[79] *But see* Martha Luerhrmann, *Circuit Board Designs*, http://www3.wcl.american.edu/cni/9604/8930.html (accessed Dec. 12, 2008) (discussion on CNI-Copyright mailing list, April, 1996).

copyright infringement in a "printed circuit board assembly."[80] Nevertheless, the court's discussion focused primarily on unauthorized copying of related software and failed to consider the issue of copyright in the circuit boards.

Given that the arrangement of components and wiring traces on a printed circuit board is subject to personal choices on the part of the designer, it is reasonable to argue that the circuit board is a work subject to copyright, but as with most of the outputs of the design process, that copyright is likely to be weak. However, a weak copyright is much better than no copyright at all, particularly where protection against literal copying of the entire board, rather than incorporation of only certain elements of the design into another work, is the goal.

IV. OPEN HARDWARE LICENSES

TAPR is a non-profit organization of amateur radio operators who are interested in advancing the state of radio technology, particularly through digital and computer-related communication techniques.[81] TAPR's primary role is to support groups of amateurs working on digital communication projects; a large part of this support involves helping these groups turn their project concepts into reproducible designs, and making these projects available as either kits or finished products to other amateurs. In 2005, TAPR began working with an informal group on the development of high performance software defined radio products ("HPSDR").[82] This group of HPSDR[83] developers wanted to contribute their time and expertise to the amateur radio community, but they did not want their work co-opted by commercial entities. They asked TAPR for assistance in developing a license that would help them achieve their goal of building a Free-Software-like community of hardware developers protected from commercial expropriation. TAPR asked the author of this article to take on this project, and the result was the TAPR Open Hardware License ("OHL").[84]

The TAPR Open Hardware License

The OHL attempts to achieve the goals of encouraging a community

---

[80] 166 F.3d 772, 790 (5th Cir. 1999).

[81] TAPR, *Organization*, http://www.tapr.org/organization.html (last updated May 26, 2007).

[82] A software defined radio is one in which many of the tasks previously performed by analog circuits using discrete components are performed on a computer using digital signal processing techniques. *See generally* Adrio Communs. Ltd., *RadioElectronics.com, Software Defined Radio, SDR*, http://www.radio-electronics.com/info/receivers/software_defined_radio/sdr.php (accessed May 16, 2009).

[83] High Performance Software Defined Radio Group, *High Performance Software Defined Radio, An Open Source Design, Project Description*, http://www.openhpsdr.org (accessed May 16, 2009).

[84] TAPR, *Publications: Open Hardware License, The TAPR Open Hardware License*, http://tapr.org/ohl.html (last updated May 26, 2007). The author attached the TAPR Open Hardware license as an appendix to this article for the convenience of the reader.

of development and discouraging expropriation of OHL designs, while taking into account the legal issues discussed above. The author of this article was its principal drafter, but he had assistance from numerous colleagues in both the legal and amateur radio communities.[85] After circulation of initial drafts among a small group of attorneys, the developers involved in the HPSDR project, and other interested amateur radio hobbyists, a working draft was publicly released. A sixty-day comment period, hosted on a web forum,[86] provided valuable feedback from the wider development community. In May 2007, TAPR formally adopted the TAPR Open Hardware License ("OHL") and the TAPR Noncommercial Hardware License ("NCL") and contributed them to the Open Source Hardware community for use by anyone. The two licenses are identical save for a provision in the NCL limiting commercial use. In the next paragraphs, references to the OHL also apply to the NCL unless specifically noted otherwise.

In the discussion that follows, the author uses certain shorthand terms for simplicity; these terms are also used in the license documents themselves. "Documentation" is the set of files and information that define a project placed under the OHL. "Products" are things manufactured based on the Documentation. The "Licensor" is any person who contributes to the Documentation, and a "Licensee" is someone who uses or modifies the Documentation, or makes Products based on it.

The OHL attempts to achieve the same ends for hardware as the GPL does for software, but follows a somewhat different route to those ends. The GPL is expressly a copyright license, and claims not to be a contract. Conversely, the OHL is built around both license and contract concepts[87] and operates similarly to a click-wrap agreement[88] in that it becomes effective through assent manifested by a party taking certain actions. It expressly sets up consideration for the mutual benefits and obligations it creates. In most areas, the OHL speaks functionally rather than legally; in other words, it describes what parties must and must not do,

---

[85] The author would especially like to thank attorneys Kirk Johnsen and Robert Lech, and radio amateurs Bruce Perens and Bdale Garbee for invaluable assistance.

[86] *See generally* Bruce Perens, *Technocrat Open Hardware License - Call for Public Review*, http://technocrat.net/d/2007/2/5/14355/index.html/ (posted Feb. 5, 2007, 6:08 PST).

[87] The OHL consciously uses license terminology to maintain simplicity and familiarity; those involved in the drafting deliberately decided to trade linguistic precision for wider understanding.

[88] A click-wrap agreement is

> [A]n agreement, formed entirely in an online environment such as the Internet, which sets forth the rights and obligations between parties. The term "click-wrap" is derived from the fact that such online agreements often require clicking with a mouse on an on-screen icon or button to signal a party's acceptance of the contract.

Francis M. Buono & Jonathan A. Friedman, *Maximizing the Enforceability of Click-Wrap Licenses*, 4 J. Tech. L. & Pol'y 3 (Fall 1999) (available at http://grove.ufl.edu/~techlaw/vol4/issue3/friedman.html).

without specifically describing the intellectual property regime underlying those obligations. Its goal is to enforce desired behavior without the limitations that a more technical approach might create.

Much of the effort (and discussion) in the OHL drafting process was directed at determining the behavior that should trigger certain obligations, particularly the patent immunity grant discussed below. Translating from the GPL's software focus to the world of tangible objects was not straightforward. In particular, determining the proper boundaries to draw around the license's obligations was an iterative process. Several OHL sections were modified as a result of the comment period to ensure that the OHL does not inadvertently extend its reach too far. The following paragraphs describe some of the OHL's key provisions.

The OHL text includes a non-binding Preamble that explains the philosophy underlying the document, and describes in simple terms how to use the OHL to protect a design. While such detailed introductions are unusual in commercial contracts, they are valuable in contexts where the license may be read and implemented by technical, rather than legal, experts.[89] One concern of the author's was ensuring that the Preamble remained useful, while not inadvertently overriding the specific terms of the agreement. Some simple language addressed this problem: "While the terms and conditions below take precedence over this preamble, here is a summary:"[90]

OHL Section 1.6 expressly disclaims applicability to software, firmware, or code loaded into programmable devices. Some commentators asked that the OHL cover both hardware designs and the software that might be executed on that hardware. However, the differences between protection of hardware designs, as described in this article, and protection of software, are so substantial that a single license attempting to protect both would be cumbersome. For this reason, the OHL limits its scope to hardware and (in its preamble) points developers to other documents, such as the GPL, for use with code.[91]

Despite statements earlier in this article that downplay the role of patents in the Open Source Hardware mindset, the OHL contains a patent provision, and this is perhaps the most unique feature of the document. The goal of Section 2 is to create a "patent-free" zone around Documentation

---

[89] The GPL also includes such an introduction. Free Software Foundation, *GNU General Public License Version 3*, http://www.gnu.org/copyleft/gpl.html (updated 29 June 2007).

[90] OHL, Preamble.

[91] There are several open source projects focused on developing code for use in Field Programmable Gate Arrays ("FPGA") and Complex Programmable Logic Devices ("CPLD") and the now almost obsolete Application Specific Integrated Circuits. These projects typically use existing Open Source licenses. *See e.g.* OpenCores, *Frequently Asked Questions*, http://www.opencores.org/?do=faq (accessed Feb. 20, 2009) (stating most contributors use GNU or BSD licenses).

and Products. It does this by requiring each Licensor,[92] and each person who makes Products, or distributes modified Documentation,[93] to grant a personal immunity from suit that follows the Documentation and Products. The immunity is in favor of every Licensee (a broad class identified in Section 1.5) as well as every possessor of Products.

These grants of immunity are important because they form part of the consideration for the agreement, based on the idea that waiving a right to bring a claim, even if that claim is uncertain, has value because it provides the recipient of the immunity with peace of mind against potential patent claims.[94] While some courts have held that the right to collect royalties under a license agreement ceases if the underlying patent is declared invalid,[95] that theory should not apply when, as here, the question is not the *validity* of an identified patent, but rather the *possible existence* of patents that might affect the Licensee or possessor.

The OHL grant of immunity is analogous to a quitclaim deed -- a document that transfers whatever ownership the grantor possesses, be it complete or none.[96] Just as the recipient of a quitclaim deed receives comfort that the grantor, at least, will not claim the conveyed property, the recipient of an immunity from suit for infringement receives comfort that the grantor, at least, will not challenge his or her right to use Documentation or Products for which the immunity was granted.

However, the immunity grant is subject to certain constraints intended to keep the OHL from becoming a trap for the unwary. For example, one who distributes Documentation without doing more (e.g., by placing it in unmodified form on a website) does not grant immunity simply by virtue of that act.[97] In addition, a manufacturer who makes Products only on behalf of someone else does not grant immunity based on that act alone; were he to make Products for his own benefit, the situation would be different.[98] Finally, the immunity does not extend to infringement arising from modifications made by others.[99]

In Section 2.2, those who "make or have Products made, or

---

[92] OHL § 2.1.

[93] OHL § 2.2.

[94] Of course, that peace of mind does not extend to third-party claims; the immunity only protects against claims by the person granting it.

[95] *See e.g. Drackett Chem. Co. v. Chamberlin Co.*, 63 F.2d 853, 855 (6th Cir. 1933) (holding that final adjudication of a patent's invalidity results in an "eviction" free the licensee from paying royalties accruing after that adjudication).

[96] A quitclaim deed is one "intended to pass any title, interest, or claim which the grantor may have in the premises, but not professing that such title is valid, nor containing any warranty or covenants for title." *Black's Law Dictionary* 1126 (5th ed., West 1979).

[97] A non-Licensor grants an immunity only if he or she makes or has Products made, or distributes Documentation that he or she modified. OHL § 2.2.

[98] OHL § 2.3.

[99] OHL §§ 2.2 and 2.3.

distribute Documentation that you have modified" grant a patent immunity. Early versions of the language did not contain the "you have modified" clause. It was added after comments raised concerns about unfairly ensnaring one who might, for example, do nothing more than post unmodified Documentation on a website, or even simply possess a Product made by someone else. The final provision makes clear that some active participation in the life of the Documentation or Product is required to trigger the immunity.

Another question arose regarding whether someone who merely possesses a Product should be deemed a "Licensee," and therefore subject to the OHL. Commentators viewed this as an excessive reach. Consequently, "Licensee" retains a narrow definition: "By (a) using, copying, modifying, or distributing the Documentation, or (b) making or having Products made or distributing them, you accept this Agreement, agree to comply with its terms, and become a "Licensee."[100]

However, the OHL's intention is to extend the immunity not just to Licensees, but to all rightful possessors of Products. In order to protect those who merely acquire a Product without being involved in its development or manufacture, Section 2.1 extends the patent immunity to "possessors" as a class distinct from "Licensees," and Section 5.2 requires that the maker distribute Documentation (or instructions on how to obtain the Documentation) along with the Products. Thus, one who a possesses a Product has the benefit of the immunity, as well as an opportunity to obtain the Documentation, but provided he or she does not do more, does not incur the obligations of a Licensee. Note that the obligation to distribute Documentation ends there and does not extend to subsequent possessors (in other words, one who possesses a Product is not required to provide or otherwise make the Documentation available if he transfers it to someone else); it was felt that an obligation to keep the Documentation together with the Product indefinitely was impractical.

Sections 3 and 4 of the OHL establish rights to modify and distribute the Documentation that are substantially similar to those of the GPL. Unlike the GPL, however, Section 3 requires that "downstream" developers email their modifications "upstream" to the earlier developers. This provision is somewhat unusual in the open source world and, although it was important to the TAPR developers,101 the "feedback" requirement

[100] OHL § 1.5.

[101] One contributor said: "My open-source DSP code, has been posted for almost 4 years and I have had essentially zero feedback. I don't know if anyone has ever used it, improved it, or ??? A feedback requirement at least lets me know no one thinks the work I did was worth their time to improve it." E-mail from Lyle Johnson to John Ackermann, *Email Feedback Requirement?* (Jan. 21, 2007, 11:30 p.m. EST). Another reason put forward for a feedback requirement was the possibility that hardware products, much more than software, have the potential to cause physical injury, and later corrections to fix safety issues should be sent upstream to help reduce the risk of injury.

was surprisingly controversial during the comment process. Some commentators viewed it as inconsistent with the Open Source model.

Originally, the language contemplated a central database where feedback would be sent and stored, but as a result of comments received, this requirement was changed to a "best efforts" attempt to send the modifications by email to all upstream developers who provided email addresses in the Documentation. An attempt to send feedback to the provided email address fulfills the downstream developer's obligation even if delivery to that address fails. This approach balances the three issues voiced by developers: feedback concerns (that developers have a way to learn of issues with their work), privacy concerns (that developers need not provide an email address if they do not wish to), and complexity concerns (that a centralized database might disappear, and that it may be burdensome to track down changed email addresses).

Finally, Section 7 contains disclaimers of warranty and liability; in common with Open Source Software licenses, the OHL attempts to protect those who give freely from suffering economic harm as a consequence. A difference from typical Open Source Software licenses is Section 7.4's indemnification provision covering defects in design, manufacture, or operation of a Product. This indemnity arose from the unique capacity of hardware to cause injury. A hardware design publisher who receives no financial reward might reasonably request indemnification from someone who implements the design, distributes it to others, and thereby causes injury. The indemnification runs only to Licensors (*e.g.*, those who have contributed to the design), and not to everyone in the chain of possession. This furthered the practical goal of protecting those who contributed to the design while not creating a broad universe of potential indemnities.

It is finally worth noting that the OHL is itself copyrighted, and that the copyright owner's exclusive rights are put to use. Like the GPL,[102] the OHL encourages widespread adoption, but attempts to control modifications to ensure that a reference to "OHL Version 1" is always a reference to the same document: "TAPR owns the copyright to the OHL, but grants permission to any person to copy, distribute, and use it in unmodified form."[103]

The TAPR Noncommercial Hardware License

A number of the HPSDR group members who sparked this drafting

[102] "Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed." Free Software Foundation, *GNU General Public License Version 3*, http://www.gnu.org/copyleft/gpl.html (updated 29 June 2007).

[103] OHL, § 6; NCL, § 6.

effort were concerned about controlling the commercial use of Documentation and Products. While early versions of the OHL attempted to accommodate this request by including optional language, ultimately their concern was addressed by creating another document, the TAPR Noncommercial Hardware License ("NCL"), which adds a commercial use restriction to the OHL. However, it is worth considering why this issue is important, as it points out another distinction between the worlds of hardware and software.

The Open Source Software community views the idea of a license limiting commercial use of a computer program as fundamentally inconsistent with the Open Source philosophy. The Open Source Definition prohibits discrimination based on field of endeavor. "The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research."[104]

However, there is a significant difference between Open Source Software and Open Source Hardware that can summed up in the saying– "electrons are cheap, but atoms are expensive." In other words, the out-of-pocket cost to develop software is virtually nil, as is the cost of making and electronically distributing copies; electrons are cheap. Conversely, a hardware project involves tangible items that must be acquired for each unit made, such as circuit boards and components; atoms are expensive. Manufacturing a few prototype circuit boards may cost more than one hundred dollars. The parts mounted on those prototype boards to test its correctness may cost several hundred dollars more. This expenditure is a small sum for a commercial enterprise but a noticeable amount to most individuals. Therefore, the initial phases of a hardware project often mean a significant investment on the part of a non-commercial developer.

If the designer's goal is to make the product available to others, the economics of production come into play. Circuit boards and components are both subject to steep quantity discounts; production quantities in the tens of millions brought PC and consumer electronics prices down to what they are today. Volume discounts for quantities as small as one hundred components are significant over the single-unit price.[105]

This implies that a marketing effort, even on a non-profit basis, requires two things: (1) printed circuit board and part orders large enough to drive down costs; and (2) an accurate count of potential buyers in order to minimize the amount of unsold inventory. Even established non-profit

---

[104] Ken Coar, *Open Source Initiative, The Open Source Definition* ¶ 6, http://opensource.org/docs/osd (posted July 7, 2006).

[105] For example, the initial prototypes of a circuit board the author designed cost approximately $45 each. When TAPR subsequently ordered 200 of those boards, the price dropped to less than $6.00.

organizations confront the issue of upfront costs. Some of the HPSDR projects supported by TAPR required a six-figure investment. Recouping that investment required selling a very large percentage of the boards and components ordered.

Given these economic facts, concern about avoiding market dilution, at least in the early days of a project, was at the forefront of discussions leading to the OHL. If the complete design package becomes available to the public before the sponsoring group's investment is recouped, others may choose to build the product themselves instead of buying one from the sponsor. The end result could be a powerful disincentive against development.

One approach to this problem, if all the developers agree, is to use a dual-license model, as is sometimes done with software products.[106] For example, the developers could license one entity to do initial production on an exclusive basis. Once those units have been sold, the developers would release the Documentation under the OHL, allowing others to modify it and manufacture the Product themselves. In fact, this was the model used for some of the HPSDR projects supported by TAPR.

While the author of this article believes that such a dual-license model is the most practical approach to address this situation, HPSDR community members asked for a license that permitted disclosure of the Documentation package, while limiting its use. The result is the TAPR Noncommercial Hardware License,[107] which is identical to the OHL but for an added Section 5.3 and a few minor changes required for consistency with that section:

> Products may only be made for your personal use or for distribution on a non-profit basis (*e.g.*, sold for no more than the actual cost of components, assembly, and shipping) Making more than ten units in any twelve month period for any purpose is deemed commercial use and is prohibited. These limitations may be altered or waived through written or email permission obtained from each Licensor.[108]

The waiver provision contained in the last sentence of Section 5.3 is intended to encourage potential manufacturers to negotiate with the Licensor to undertake production on terms that meet the Licensor's goals. Because of

---

[106] *See e.g.* Nokia Corp., *Qt Licensing, The Qt Licensing Model*, http://www.qtsoftware.com/products/appdev/licensing (accessed Feb. 20, 2009) (Qt libraries are subject to different licenses depending on whether the user will incorporate QT into a proprietary, or an open source product).

107 TAPR, *Publications: Noncommercial Hardware License, The TAPR Noncommercial Hardware License*, http://www.tapr.org/NCL (last updated May 26, 2007).

[108] NCL, § 5.3.

the relatively small number of Licensors involved in a typical Open Source Hardware project, this is potentially a practical approach. While TAPR has made the NCL available to the community on the same basis as the OHL, it encourages developers to use the OHL if possible.

V. CONCLUSION

The idea of Open Source Hardware is not new; it dates back to the earliest days of electronics when the only way to obtain a radio was to build one from plans published in books and magazines. However, when products were built by hand using point-to-point wiring techniques, the intellectual property issues raised were straightforward; no one questioned whether a chassis full of wires was a derivative of the schematic diagram. By contrast, today's development process for electronic products, particularly related to printed circuit boards, opens the door to numerous intellectual property questions.

As old models of hardware development become impractical due to complex yet almost microscopic components, EDA tools and on-demand circuit board manufacture make it possible for designers outside the traditional electronics industry to use these new capabilities to produce state-of-the-art designs. Since much hardware today is connected with some sort of computer code -- whether software running in a microprocessor or firmware running in a logic device – many of those designers are familiar with Open Source in the software domain and seek to build similar concepts of community and sharing in a hardware development model.

The TAPR Open Hardware and Noncommercial Hardware Licenses are a first attempt to translate the concepts of Open Source Software into an Open Source Hardware community. Certainly, others will follow, and hopefully they will be able to build on this early work. [109]

[109] The TAPR Open Hardware License is not the only effort to develop an Open Source Hardware model, although it appears to be the first such license designed for general use and not tightly tied to a specific hardware project. The Balloon License in 2003 was one early attempt. The Balloon Project, *Balloon License*, http://www.balloonboard.org/licence.html (accessed May 16, 2009). The Balloon group appears to be working on an updated version of that license, *see* http://balloonboard.org/balloonwiki/OpenHardwareLicense, but the status of that work is currently unknown. The Open Hardware Foundation hosts a mailing list about development of an Open Hardware License; as of April 8, 2009, there had been no activity on that mailing list since September, 2008. Open Hardware Foundation, *The ohf-licenses Archive*, http://mail.openhardwarefoundation.org/pipermail/ohf-licenses_openhardwarefoundation.org/ (accessed May 16, 2009). Additionally, the author was recently asked to develop a version of the OHL that would address partial incorporation of an Open Hardware circuit design into an otherwise proprietary circuit, along the lines of the Library General Public License (LGPL) published by the FSF to allow incorporation of Open Source Software libraries into closed-source products under certain conditions

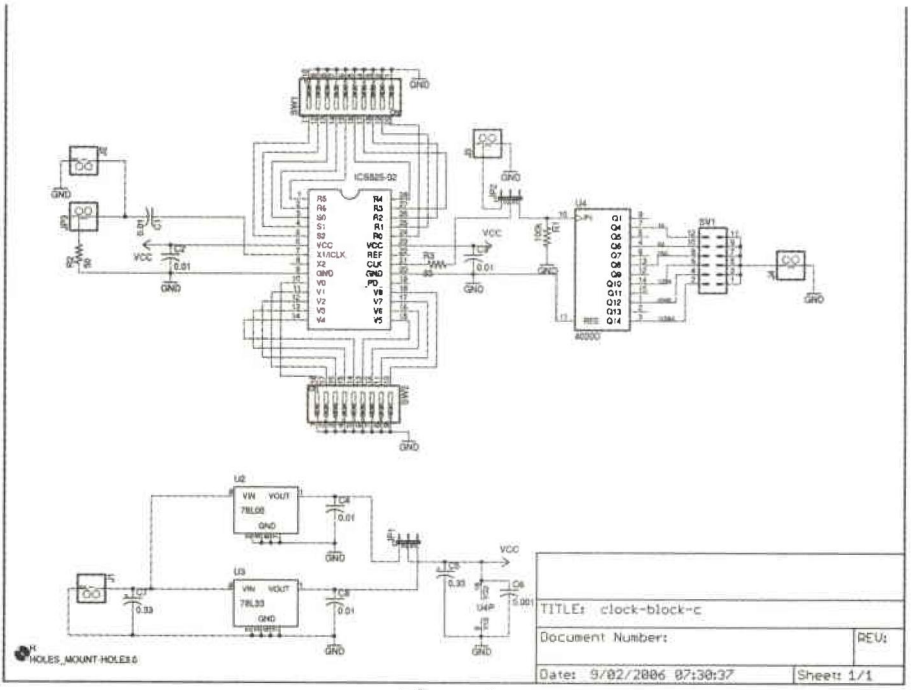

Figure 1

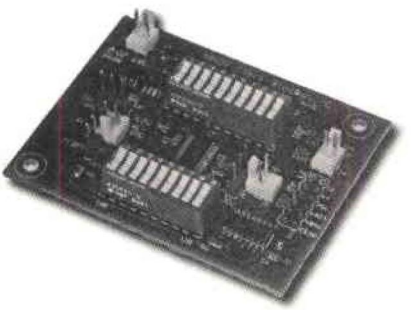

Figure 2

Net	Part	Pin
N$20	C8	2
	IC1	3
	R1	1
	R8	2
N$21	L1	1
	R21	2

Figure 3

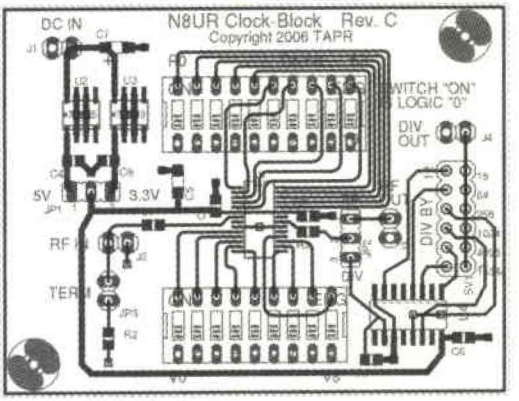

Figure 4

```
G75*
G70*
%OFA0B0*%
%FSLAX24Y24*%
%IPPOS*%
%LPD*%
%AMOC8*
5,1,8,0,0,1.08239X$1,22.5*
%
%ADD10O,0.0600X0.1200*%
%ADD11O,0.1200X0.0600*%
%ADD12O,0.0560X0.1120*%
%ADD13O,0.0520X0.1040*%
%ADD14OC8,0.0600*%
%ADD15O.0.1120X0.0560*%
```

Figure 5

APPENDIX

The TAPR Open Hardware License

Version 1.0 (May 25, 2007)

PREAMBLE

Open Hardware is a thing – a physical artifact, either electrical or mechanical – whose design information is available to, and usable by, the public in a way that allows anyone to make, modify, distribute, and use that thing. In this preface, design information is called "documentation" and things created from it are called "products."

The TAPR Open Hardware License ("OHL") agreement provides a legal framework for Open Hardware projects. It may be used for any kind of product, be it a hammer or a computer motherboard, and is TAPR's contribution to the community; anyone may use the OHL for their Open Hardware project. You are free to copy and use this document provided only that you do not change it.

Like the GNU General Public License, the OHL is designed to guarantee your freedom to share and to create. It forbids anyone who receives rights under the OHL to deny any other licensee those same rights to copy, modify, and distribute documentation, and to make, use and distribute products based on that documentation.

Unlike the GPL, the OHL is not primarily a copyright license. While copyright protects documentation from unauthorized copying, modification, and distribution, it has little to do with your right to make, distribute, or use a product based on that documentation. For better or worse, patents play a significant role in those activities. Although it does not prohibit anyone from patenting inventions embodied in an Open Hardware design, and of course cannot prevent a third party from enforcing their patent rights, those who benefit from an OHL design may not bring lawsuits claiming that design infringes their patents or other intellectual property.

The OHL addresses unique issues involved in the creation of tangible, physical things, but does not cover software, firmware, or code loaded into programmable devices. A copyright-oriented license such as the GPL better suits these creations.

How can you use the OHL, or a design based upon it? While the terms and conditions below take precedence over this preamble, here is a summary:

- You may modify the documentation and make products based upon it.
- You may use products for any legal purpose without limitation.

- You may distribute unmodified documentation, but you must include the complete package as you received it.
- You may distribute products you make to third parties, if you either include the documentation on which the product is based, or make it available without charge for at least three years to anyone who requests it.
- You may distribute modified documentation or products based on it, if you:
  - License your modifications under the OHL.
  - Include those modifications, following the requirements stated below.
  - Attempt to send the modified documentation by email to any of the developers who have provided their email address. This is a good faith obligation – if the email fails, you need do nothing more and may go on with your distribution.
- If you create a design that you want to license under the OHL, you should:
  - Include this document in a file named LICENSE (with the appropriate extension) that is included in the documentation package.
  - If the file format allows, include a notice like "Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)" in each documentation file. While not required, you should also include this notice on printed circuit board artwork and the product itself; if space is limited the notice can be shortened or abbreviated.
  - Include a copyright notice in each file and on printed circuit board artwork.
  - If you wish to be notified of modifications that others may make, include your email address in a file named "CONTRIB.TXT" or something similar.
- Any time the OHL requires you to make documentation available to others, you must include all the materials you received from the upstream licensors. In addition, if you have modified the documentation:
  - You must identify the modifications in a text file (preferably named "CHANGES.TXT") that you include with the documentation. That file must also include a statement like "These modifications are licensed under the TAPR Open Hardware License."

- You must include any new files you created, including any manufacturing files (such as Gerber files) you create in the course of making products.
- You must include both "before" and "after" versions of all files you modified.
- You may include files in proprietary formats, but you must also include open format versions (such as Gerber, ASCII, Postscript, or PDF) if your tools can create them.

- **TERMS AND CONDITIONS**

1. Introduction

1.1 This Agreement governs how you may use, copy, modify, and distribute Documentation, and how you may make, have made, and distribute Products based on that Documentation. As used in this Agreement, to "distribute" Documentation means to directly or indirectly make copies available to a third party, and to "distribute" Products means to directly or indirectly give, loan, sell or otherwise transfer them to a third party.

1.2 "Documentation" includes:

(a) schematic diagrams;

(b) circuit or circuit board layouts, including Gerber and other data files used for manufacture;

(c) mechanical drawings, including CAD, CAM, and other data files used for manufacture;

(d) flow charts and descriptive text; and

(e) other explanatory material.

Documentation may be in any tangible or intangible form of expression, including but not limited to computer files in open or proprietary formats and representations on paper, film, or other media.

1.3 "Products" include:

(a) circuit boards, mechanical assemblies, and other physical parts and components;

(b) assembled or partially assembled units (including components and subassemblies); and

(c) parts and components combined into kits intended for assembly by others;

which are based in whole or in part on the Documentation.

1.4 This Agreement applies to any Documentation which contains a notice stating it is subject to the TAPR Open Hardware License, and to all Products based in whole or in part on that Documentation. If Documentation is distributed in an archive (such as a "zip" file) which includes this document, all files in that archive are subject to this Agreement unless they are specifically excluded. Each person who contributes content to the Documentation is referred to in this Agreement as a "Licensor."

1.5 By (a) using, copying, modifying, or distributing the Documentation, or (b) making or having Products made or distributing them, you accept this Agreement, agree to comply with its terms, and become a "Licensee." Any activity inconsistent with this Agreement will automatically terminate your rights under it (including the immunities from suit granted in Section 2), but the rights of others who have received Documentation, or have obtained Products, directly or indirectly from you will not be affected so long as they fully comply with it themselves.

1.6 This Agreement does not apply to software, firmware, or code loaded into programmable devices which may be used in conjunction with Documentation or Products. Such software is subject to the license terms established by its copyright holder(s).

2. Patents

2.1 Each Licensor grants you, every other Licensee, and every possessor or user of Products a perpetual, worldwide, and royalty-free immunity from suit under any patent, patent application, or other intellectual property right which he or she controls, to the extent necessary to make, have made, possess, use, and distribute Products. This immunity does not extend to infringement arising from modifications subsequently made by others.

2.2 If you make or have Products made, or distribute Documentation that you have modified, you grant every Licensor, every other Licensee, and every possessor or user of Products a perpetual, worldwide, and royalty-free immunity from suit under any patent, patent application, or other intellectual property right which you control, to the extent necessary to make, have made, possess, use, and distribute Products. This immunity does not extend to infringement arising from modifications subsequently made by others.

2.3 To avoid doubt, providing Documentation to a third party for the sole purpose of having that party make Products on your behalf is not considered "distribution," and a third party's act of making Products solely on your behalf does not cause that party to grant the immunity described in the preceding paragraph.

2.4 These grants of immunity are a material part of this Agreement, and form a portion of the consideration given by each party to the other. If any court judgment or legal agreement prevents you from granting the immunity required by this Section, your rights under this Agreement will terminate and you may no longer use, copy, modify or distribute the Documentation, or make, have made, or distribute Products.

3. Modifications

You may modify the Documentation, and those modifications will become part of the Documentation. They are subject to this Agreement, as are Products based in whole or in part on them. If you distribute the modified Documentation, or Products based in whole or in part upon it, you must email the modified Documentation in a form compliant with Section 4 to each Licensor who has provided an email address with the Documentation. Attempting to send the email completes your obligations under this Section and you need take no further action if any address fails.

4. Distributing Documentation

4.1 You may distribute unmodified copies of the Documentation in its entirety in any medium, provided that you retain all copyright and other notices (including references to this Agreement) included by each Licensor, and include an unaltered copy of this Agreement.

4.2 You may distribute modified copies of the Documentation if you comply with all the requirements of the preceding paragraph and:

(a) include a prominent notice in an ASCII or other open format file identifying those elements of the Documentation that you changed, and stating that the modifications are licensed under the terms of this Agreement;

(b) include all new documentation files that you create, as well as both the original and modified versions of each file you change (files may be in your development tool's native file format, but if reasonably possible, you must also include open format, such as Gerber, ASCII, Postscript, or PDF, versions);

(c) do not change the terms of this Agreement with respect to subsequent licensees; and

(d) if you make or have Products made, include in the Documentation all elements reasonably required to permit others to make Products, including Gerber, CAD/CAM and other files used for manufacture.

5. Making Products

5.1 You may use the Documentation to make or have Products made, provided that each Product retains any notices included by the Licensor (including, but not limited to, copyright notices on circuit boards).

5.2 You may distribute Products you make or have made, provided that you include with each unit a copy of the Documentation in a form consistent with Section 4. Alternatively, you may include either (i) an offer valid for at least three years to provide that Documentation, at no charge other than the reasonable cost of media and postage, to any person who requests it; or (ii) a URL where that Documentation may be downloaded, available for at least three years after you last distribute the Product.

6. NEW LICENSE VERSIONS

TAPR may publish updated versions of the OHL which retain the same general provisions as the present version, but differ in detail to address new problems or concerns, and carry a distinguishing version number. If the Documentation specifies a version number which applies to it and "any later version", you may choose either that version or any later version published by TAPR. If the Documentation does not specify a version number, you may choose any version ever published by TAPR. TAPR owns the copyright to the OHL, but grants permission to any person to copy, distribute, and use it in unmodified form.

7. WARRANTY AND LIABILITY LIMITATIONS

7.1 THE DOCUMENTATION IS PROVIDED ON AN "AS-IS" BASIS WITHOUT WARRANTY OF ANY KIND, TO THE EXTENT PERMITTED BY APPLICABLE LAW. ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND TITLE, ARE HEREBY EXPRESSLY DISCLAIMED.

7.2 IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL ANY LICENSOR BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, OR EXEMPLARY DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, THE DOCUMENTATION OR PRODUCTS, INCLUDING BUT NOT LIMITED TO CLAIMS OF INTELLECTUAL PROPERTY INFRINGEMENT OR LOSS OF DATA, EVEN IF THAT PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7.3 You agree that the foregoing limitations are reasonable due to the non-financial nature of the transaction represented by this Agreement, and

acknowledge that were it not for these limitations, the Licensor(s) would not be willing to make the Documentation available to you.

7.4 You agree to defend, indemnify, and hold each Licensor harmless from any claim brought by a third party alleging any defect in the design, manufacture, or operation of any Product which you make, have made, or distribute pursuant to this Agreement.