CSchem

Electronic circuit layout for scientists and other part-time electrical engineers

Daniel A. Wagenaar

Chapter 1

Introduction

This document describes the installation and usage of “CSchem,” a (hopefully) easy-to-use program for drawing electronic circuit diagrams written by Daniel Wagenaar. A companion program, “CPCB,” is provided for instantiating circuits on a PCB. This program is documented separately.

1.1 Why use CSchem?

There are any number of software packages and online options available that allow you to draw circuit diagrams. So why should you choose CSchem? CSchem is for you if:

- You like your circuit diagrams to look ready for publication straight from design;
- You like concentrating on the principles of your circuit while drawing the design (and leave the specific choices of components until the next day);
- You like to have quick access to the most commonly used symbols;
- You either don’t need more uncommon symbols or are willing to draw them in an SVG editor;
- You like the companion program CPCB for laying out PCBs.

On the other hand, CSchem may not be for you if:

- You need to draw very large circuits that span multiple sheets;
- You need to specify lots of parameters with your components for an automated layout workflow. (CSchem will allow you to specify component values or part numbers, but does not have specific fields for vendors, packaging information, etc.)

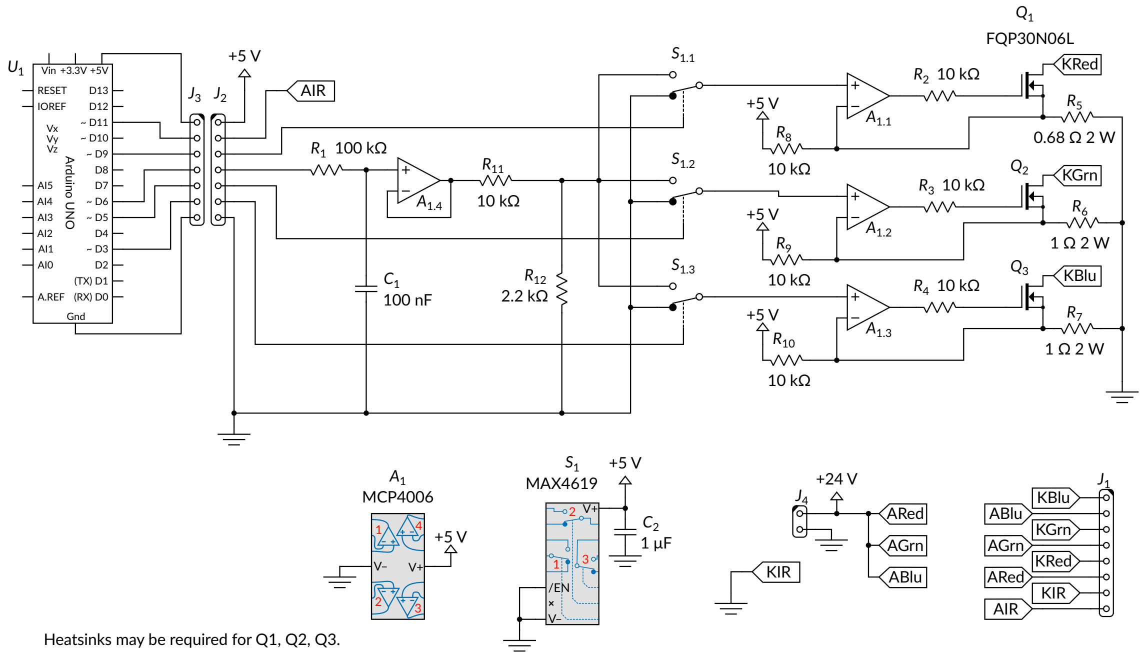


Figure 1.1: A design for an Arduino-controlled triple linear LED driver as an example of a circuit drawn with CSchem.

- You need to have access to a large library of predrawn components.
- You need a help desk on call.

1.1.1 A note on development

CSchem is being developed by an active research scientist. Practically, that means two things: On the positive side, it means that I have a vested interest in fixing bugs and improving CSchem, because I use it regularly. On the negative side, that means that, by and large, new features are added only when I need them and bugs are fixed when I have time. I certainly do welcome feature requests, but I cannot guarantee that they will get implemented quickly or at all. (If you are in a hurry, I will consider (paid) consultancy related to CSchem.) Finally, I definitely welcome contributions to either the code or the documentation. I would be very happy if CSchem turned into a community-supported open source project.

1.2 Features

CSchem circuit designs consist of a single, conceptually infinitely large sheet containing "elements" and "connections."

Elements are things like resistors and opamps, connections are simple wires connecting between pins of elements. Each element has up to two pieces of text associated with it: a circuit "reference" (e.g., "R_1," "J_2," or "$A_{3.2}$") and a "part/value" designation. The "part/value" designation is free-form. You can use it for a resistor value (e.g., "10 kΩ" or "1 Ω 3 W") or for a part number (e.g., "OPA2350") or for an arbitrary label.

Connections are wires between (pins of) elements. Where wires meet, a "junction" symbol is automatically inserted. Of course, wires can also cross without electrical contact. Connections normally are constrained to run horizontally or vertically with right-angle elbows.

The only graphical element that CSchem supports other than elements and connections are arbitrary textual annotations that can be placed anywhere on the sheet.

At present, CSchem does not have explicit support for buses with multiple signal wires or for splitting a drawing across multiple sheets.

1.3 Contacting the author

If you like CSchem or find fault with it, if you discover a bug or have a suggestion for a new feature, if you are interested in improving this documentation or have a patch to contribute to the code, I want to hear from you. My contact information is at http://www.danielwagenaar.net. I very much look forward to hearing from you. I realize that this guide is extremely terse, and I really do welcome questions, particularly if they help me to improve CSchem or its documentation.

Pasadena, January 2019

Chapter 2

Installation

The latest version of the software can always be downloaded from http://www.danielwagenaar.net/cschem.

2.1 Precompiled binaries

Binary packages for Linux, Windows and Mac OS X will be provided as time permits. You can help focus my attention on binaries simply by expressing an interest.

2.2 Compiling the source

To compile the source, start from the provided "cschem.tar.gz" archive or check out the git source at http://github.com/wagenadl/cschem. Compilation requires "Qt" version 5.6 or later.

2.2.1 Compiling on Linux or Mac OS

You will need a C++ compiler and "make". On Ubuntu Linux, this is as simple as "sudo apt-get install g++ make". On Mac OS, you need the "Command Line tools for XCode" from the Apple Developers' web site[1].

Open a terminal and "cd" to the root of the unpacked source archive. Then type "make" and fetch a cup of tea. Then, either manually copy the file "build/cschem/cschem" to some location on your PATH, or type "sudo make install" to install into "/usr/local/bin".

2.2.2 Compiling on Windows

Details to follow. Again, feel free to ask!

[1]https://developer.apple.com/xcode.

Chapter 3

Using CSchem

CSchem has a deliberately sparse user interface that may take a little getting used to. It is the author's hope, however, that users will quickly get to appreciate the simplicity of the system.

3.1 Placing elements

To place an element with a predefined symbol, simply drag the symbol from the sidebar onto the design. To place custom elements, drag in their SVG file from a Filer window.[1] (For more on custom elements, see section 4.3.) Elements can be moved around simply by dragging the mouse. To delete an element, simply hover over it and press "Delete."

3.2 Placing connections

Connections can be placed starting at pins of elements simply by dragging the mouse. The connection will start in the direction of initial motion, so drag away from the element to avoid tangles. Click to fixate elbows in the connection; click on a pin of an element or on another connection to complete the connection. Press "Return" to terminate a partially drawn connection and leave it dangling. Press "Escape" or "Delete" to abandon a partially drawn connection and remove it. To start drawing a new connection starting from an existing connection, hold "Control" and hover over the old connection. A transient marker will appear just as when you hover over a pin, and you can drag out a new connection from that point.

Existing connections can be moved and reshaped simply by dragging the mouse. While moving, connections will "snap" to nearby anchors such as pins. This normally

[1]E.g., Gnome Files in Linux, the Finder in Mac OS, or the File Explorer in Windows.

makes it easier to avoid spurious short zigzags. When snapping gets in your way, simply hold “Control” to disable it.

Sometimes, connections and up with several unnecessary elbows. To simplify a connection, double click on a segment that you would like to go away. This merges the segment with the nearest segment that runs in the same direction, eliminating also the intermediate short segment that runs in the perpendicular direction.

To delete a connection, simply hover over it and press “Delete.” This only deletes the highlighted segment, but “Control”+“B” deletes all dangling connections, so it is easy to remove the rest.

3.3 Selections and copy and paste operations

Elements can be selected either by clicking on them or by dragging an area around them. To add to an existing selection, hold “Shift” while clicking or dragging. Connections cannot be explicity selected; instead, connections between pairs of selected elements are implicitly selected. Use “Control”+“C” to copy a selection to the clipboard or “Control”+“X” to cut a selection to the clipboard. “Control”+“V” pastes the contents of the clipboard to the mouse position and selects what was pasted, so that it easy to fine position it. Normally, when dragging a selection to a new location, junctions will be automatically inserted as needed.

3.4 Adding text

Reference and part/value labels can be added to elements that don’t already display them by double clicking on the element. These labels will move with the element and can be micropositioned by dragging with the mouse. To hide a label, hover over it with the mouse pointer and press “Backspace.” (This doesn’t actually delete the annotation; it can be made visible again by double clicking the parent element.)

Arbitrary textual annotations can be placed anywhere on the canvas simply by double clicking on the canvas. At present, special formatting is not supported, but your requests will be considered. To remove an annotation, simply delete all the text in it. (Press “Control”+“A” then “Backspace” or “Delete.”)

3.5 Exporting and printing

CSchem does not directly talk to printers. However, it can export the circuit diagram as vector graphics (SVG). In addition, the parts list can be exported as a CSV file. For further convenience, a bitmap image of the diagram can be copied to the system clipboard for direct inclusion in, e.g., an electronic lab notebook. Likewise, the parts list can be copied to the system clipboard.

Chapter 4

Advanced topics

4.1 Parts list

To open a parts list as a side panel, use the “View” menu or press “Control”+“Shift”+“L”. The parts list can be used to conveniently modify the “reference” and “part/value” text associated with different elements, and allows you to add arbitrary notes to any elements. (Those notes are not displayed on the canvas.)

4.2 More about elements

It is conceptually useful to distinguish between several kinds of elements:

Ports These are nonphysical entities such as a ground reference or markers to give names to signal traces.

Parts These are mostly straightforward physical entities such as resistors, transistors, and connectors, but also more complex entities like logic gates and opamps.

Containers These are the physical devices that contain one or more logic gates or opamps.

A few examples (Figure 4.1) should help explain this. Ports and simple parts are straightforward enough (Fig. 4.1A and B). Virtual parts (Fig. 4.1C) and containers (Fig. 4.1D) may need some explanation.

Imagine the simple amplifier circuit for a photodiode in Fig. 4.2. In this circuit, V_{out} is related to the photocurrent I_{D} by $V_{\text{out}} = R_1\,I_{\text{D}}$. (To see this, consider that light hitting a photodiode induces a photocurrent to run from the cathode to the anode, i.e., in the reverse direction of the normal diode current. Because the opamp A_1 has near-infinite input impedance, that current can only run through R_1, which must therefore (Ohm’s law) develop a voltage $V = I_{\text{D}}\,R_1$.) Drawing this this circuit in this simple form is

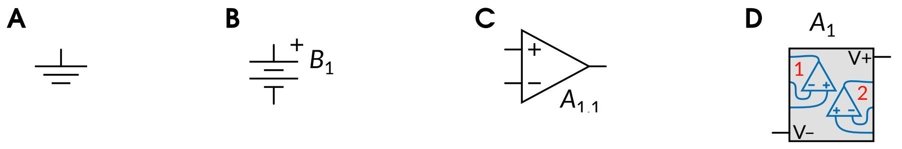


Figure 4.1: Several kinds of elements. **A.** A ground reference as an example of a port. **B.** A battery as an example of a simple part. **C.** An opamp as an example of a "virtual part." **D.** A "container" for two opamps.

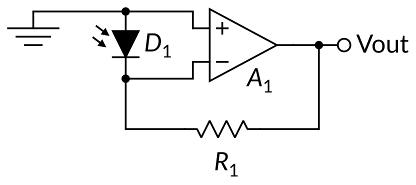


Figure 4.2: A simple photodiode amplifier.

attractive for didactic purposes: including the power connections to the opamp would make it harder to understand. However, if we are going to actually build this circuit, we do need the opamp to be powered. Rather than complicate the simple drawing by drawing power connections to the opamp symbol, however, I prefer to relegate that housekeeping stuff to a separate part of the diagram (as in Fig. 4.3). That way, the

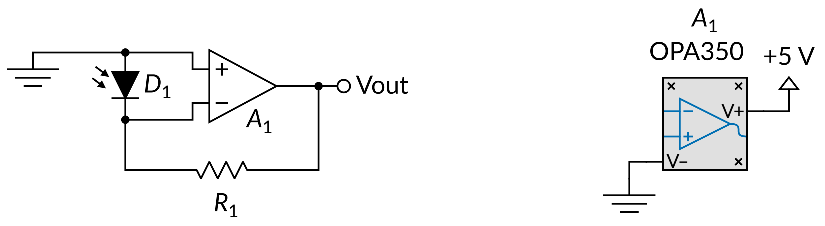


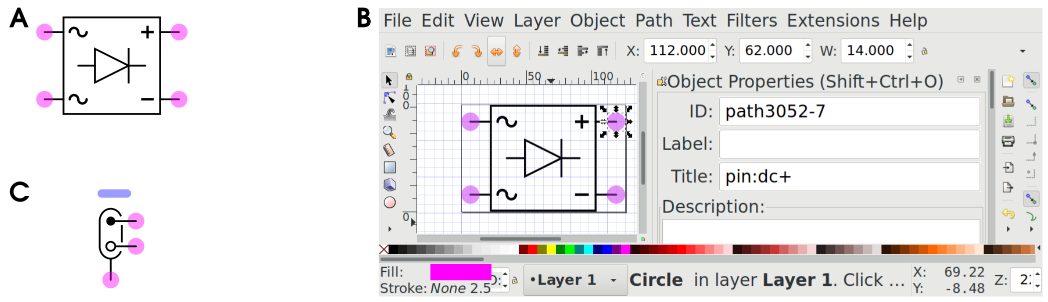
Figure 4.3: The photodiode amplifier with power supply connections.

boring stuff (such as the fact that we are using a 5-V power supply and an OPA350 opamp) does not get in the way of the interesting stuff. Note that the virtual opamp and the container are both labeled A_1, because they are ultimately one and the same when it comes to actually building the circuit on a PCB.

4.3 Drawing custom elements

CSchem shows the most commonly used circuit elements in a side bar. It also ships with a folder of less commonly used elements which can be used in a drawing simply by dragging them in from a Filer window. If you need symbols that are not in that collection, you can draw your own in an external SVG editor like Inkscape. The easiest way to begin is to load one of the symbols from the supplied folder, save it under a new name, and make edits.[1]

To make CSchem understand the structure of your file, it should contain one single group that has all the graphics of your symbol. In addition, the file should contain several pink circles[2] to mark pins. These circles should not be part of the group, but exist as separate top-level objects. Each of these circles should have a *title* tag with a specific format that identifies it as a marker for a pin location. As an example, consider a custom symbol for a 4-diode rectifier (Fig. 4.4). The title tag should have


A
B
C
File Edit View Layer Object Path Text Filters Extensions Help
Object Properties (Shift+Ctrl+O)
Fill:
Stroke: None 2.5
•Layer 1

X: 112.000 Y: 62.000 W: 14.000

ID: path3052-7
Label:
Title: pin:dc+
Description:

Circle in layer **Layer 1**. Click ...

Figure 4.4: A custom symbol for a rectifier circuit. **A.** The symbol. **B.** Inkscape screenshot showing the "title" tag on one of the pin markers.
C. Example of a custom element with a placeholder for reference text.

the form "pin:*name*" where *name* is an arbitrary text to identify the pin. Pin names should be chosen to reflect the function of a pin rather than the number of a pin in any particular physical device that implements the symbol. For instance, for a MOSFET, appropriate pin names would be "G," "D," and "S" (for Gate, Source and Drain) rather than "1," "2," and "3". If one particular numbering scheme is prevalent, it is possible to use both numbers and a name in the title tag. For instance: "pin:1/G" or "pin:3/S."

The pink circles will not appear in CSChem; they are just to mark the pin positions.

[1]I have found that starting from scratch in a new Inkscape file tends to cause problems with scaling and translation. This is at least partly due to a subtle bug in CSchem's handling of SVG files which may be fixed in a future version. For now, starting from an existing file rather than copying and pasting from an existing file into a new file is the most practical solution.

[2]The color is not actually significant; I use pink by convention. You may not, however replace the circle by some other shape.

In addition to circles that represent pins, custom symbols may also contain rectangles (conventionally with rounded corners) as placeholders for annotations such as reference text (Fig. 4.4C). These should have “annotation:ref” or “annotation:value” as their title tag.[1] If no placeholders for annotations are included, annotations will be placed at a default location.

Custom symbols that represent containers are slightly more complex (Fig. 4.5). In

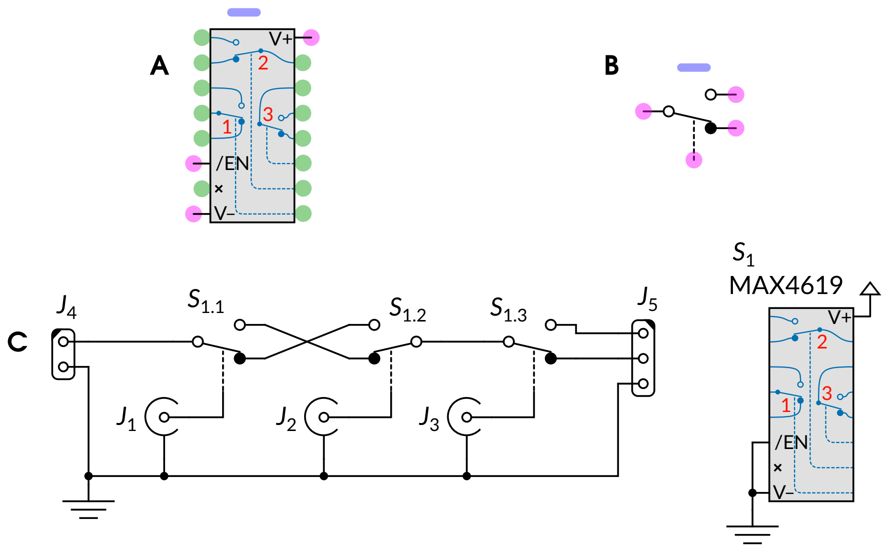


Figure 4.5: A custom symbol for a container for three electronic switches (**A**) and a custom symbol for one of those switches (**B**). Pink circles on the container symbol represent pins that should be wired on the container symbol; green circles represent pins that should be wired on the contents instead. **C.** A (rather silly) circuit demonstrating the use of these symbols.

addition to the usual pink circles, such symbols should contain green circles to represent the pins that will be linked to the contents of the container. These green circles must be titled “cp:*number*/*index*.*name*,” where *number* is the physical pin number on the standard imoplementation of the container; *index* enumerates the contained items, and *name* identifies the pin on the contained item. For example, the switch drawn in Fig. 4.5B are titled “pin:1/nc,” “pin:2/com,” “pin:3/no,” and “pin:4/sw.” (For many physical switches, pin order is “normally closed”, “common”, “normally open”; hence the numbers.) Since the container has the “common” terminal of the first contained switch as physical pin 4 (counting from top-left as is conventional for a DIP IC), the green circle by that pin is titled “cp:4/1.com.” CSchem automatically matches this to the pin named “com” on the contained element, i.e, the circle titled “pin:2/com.”

[1]As an alias for “annotation:ref,” “annotation:name” is also accepted.

(The number (“2”) is ignored for the purpose of this matching.) Likewise, physical pin 11 is the switch terminal of the third contained switch, and is therefore titled “cp:11:3/sw.” Physical pins that have no functional connections, such as number 7 in the example, can be titled “cp:7/nc.” This is not important for CSchem, but it tells the companion program CPCB not to expect a connection to that pin.

It is critical that the above conventions are followed exactly. Otherwise, the symbol will not load correctly, most likely without even an error message. (I plan to improve that situation in a future version.)

For visual consistency, the graphics of the symbol should be drawn in black lines using the same line style (1.5-px wide, solid) and grid spacing (7-px) as well as font (Lato). The insides of container elements may be drawn in other colors. It may be helpful to copy bits and pieces from several symbols to your new symbol, but remember to create your new symbol by editing an existing file rather than starting with an empty SVG.

Chapter 5

Conclusion

I hope that CSchem will be useful to you. As mentioned before, CSchem is in active development, and I am very interested in your thoughts for improvement.