Examples: Version 11.2

This is a collection of diagrams the author has had occasion to produce using m4 circuit macros and others, and dpic or gpic.

Some lists of elements from the manual Circuit_macros.pdf are included. Producing diagrams starting from a list of elements is like writing poetry starting from a list of words, so a variety of small and medium circuits and other diagrams is also included here in the expectation that you might wish to adapt some of them to your purposes. There may be other or better m4 or pic constructs for producing the same drawings, but names of the actual source-files are shown for reference.

Some of the later examples test the boundaries of what can be done when employing a "little language" like pic. Most of the examples can be processed using either dpic -p, dpic -g, or, with exceptions, gpic -t, but the possibility of other postprocessing has meant that sometimes the source is slightly more complicated than it would be if only one workflow had been assumed. The most simplicity and elegance are achieved by sticking to one pic interpreter and one postprocessor.

Color and other embellishments are not included in the standards documents for circuit elements but examples of their use to call attention to particular elements are included.

The source files can be processed individually. To produce .pdf from *file*.m4, for example, type "make *file*.pdf".

When viewing the results on-screen, be aware that there can be subtle differences in appearance with different viewers, particularly in the weight of thin lines.

Type "make" in the extras directory to see more.

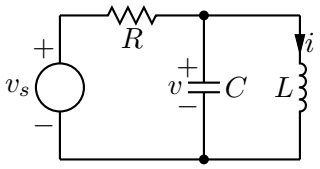


Figure 1: The quick-start example from the manual `[quick.m4]`.

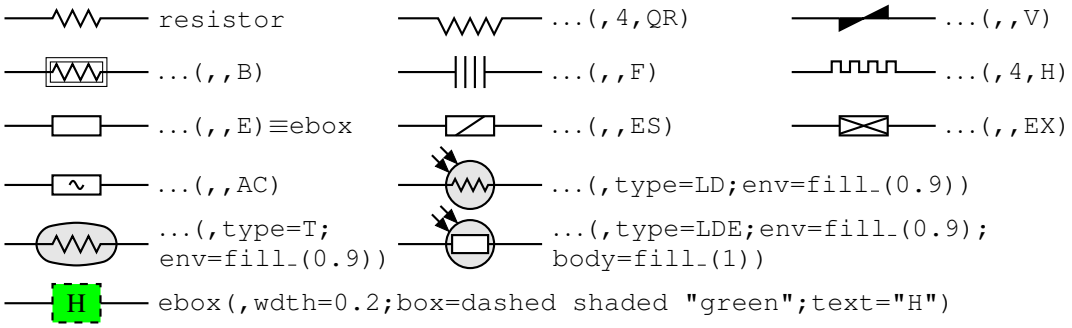


Figure 2: Resistors, showing some variations and the ebox `[Resistors.m4]`.

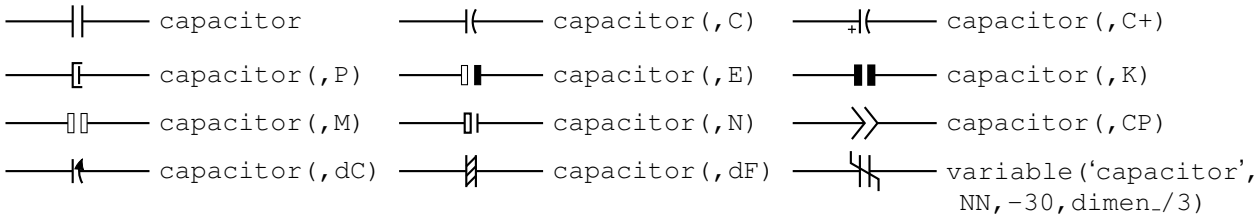


Figure 3: Capacitors `[Capacitors.m4]`.

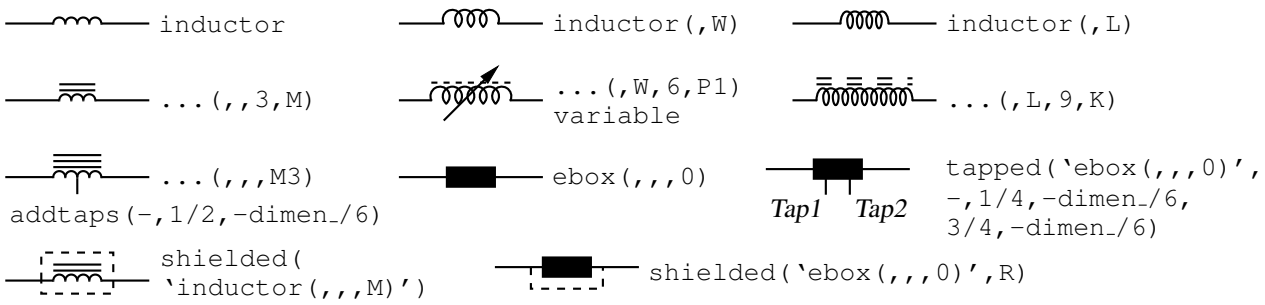


Figure 4: Inductors `[Inductors.m4]`.

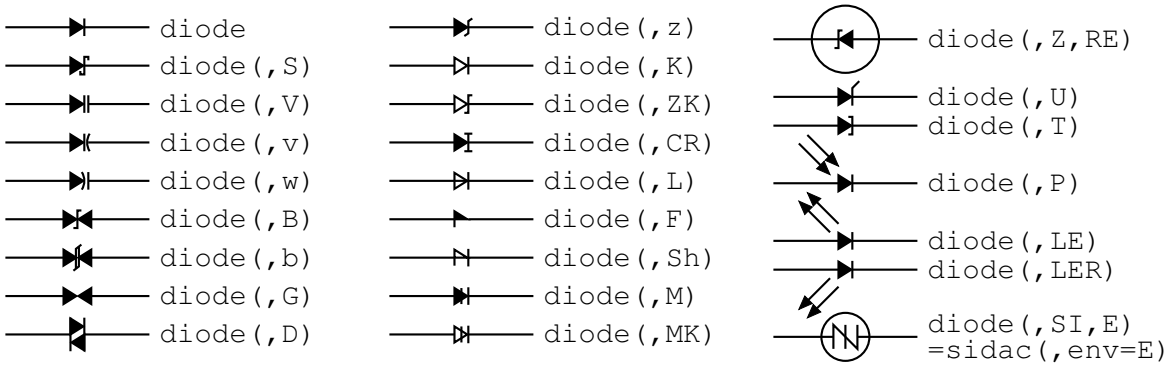


Figure 5: Diodes: appending a `K` to the second argument draws an open arrowhead `[Diodes.m4]`.

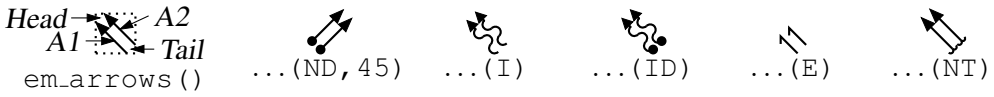


Figure 6: Radiation arrows `[Emarrows.m4]`.

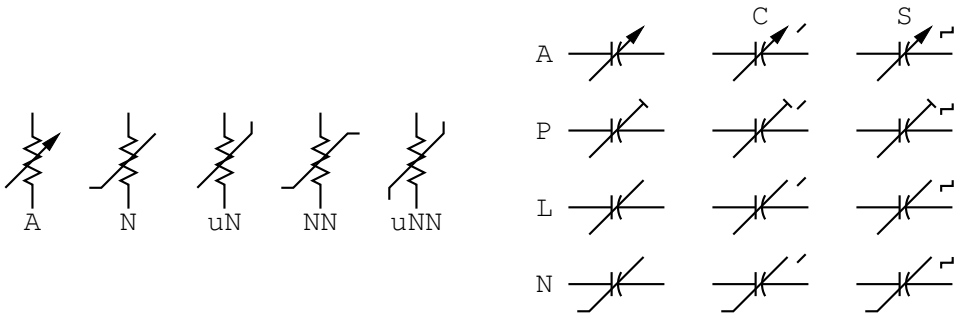


Figure 7: Arrows and marks for showing variability `[Variable.m4]`.

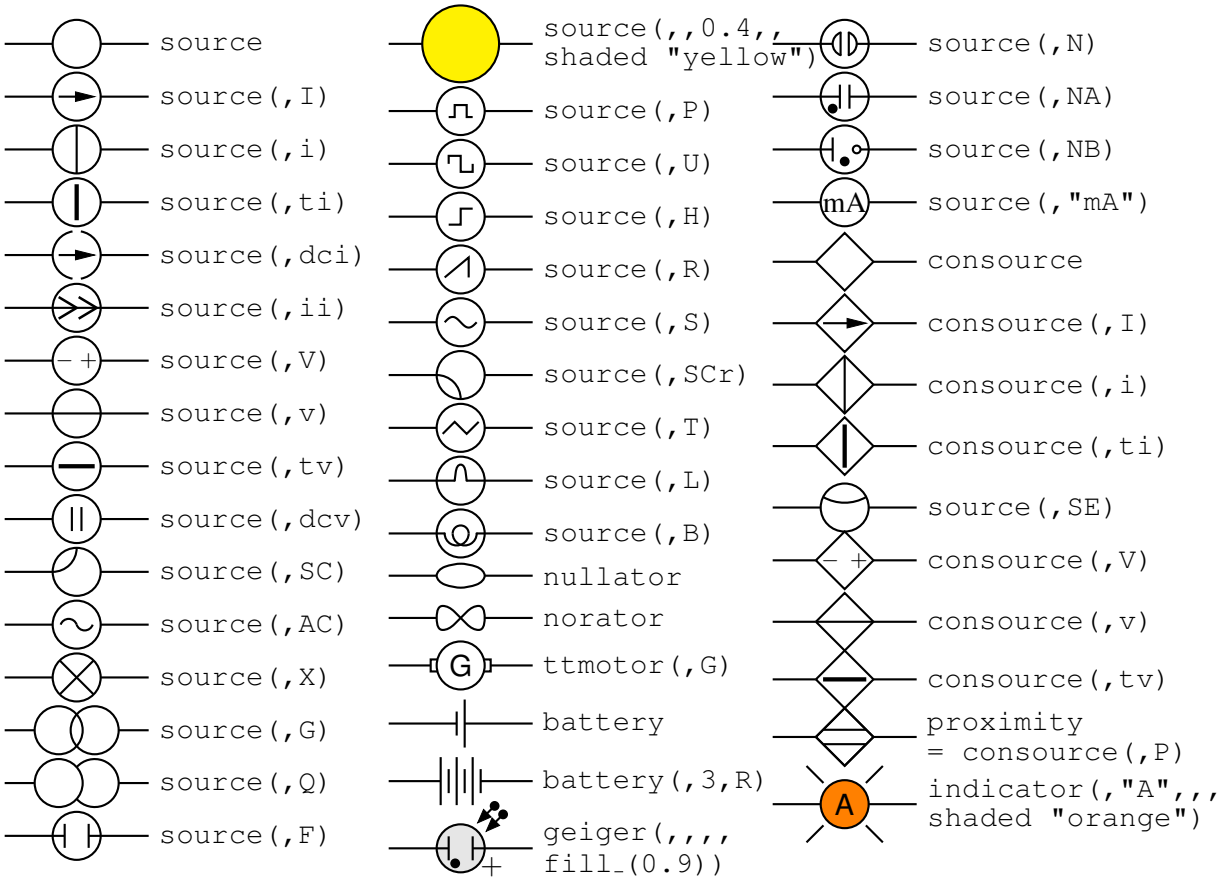


Figure 8: Sources and source-like elements `[Sources.m4]`.

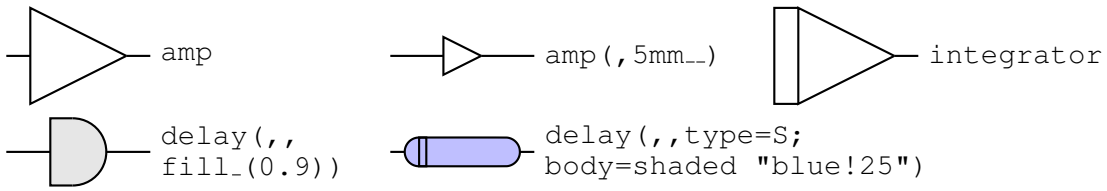


Figure 9: Macros `amp`, `delay`, and `integrator` `[AmpTable.m4]`.

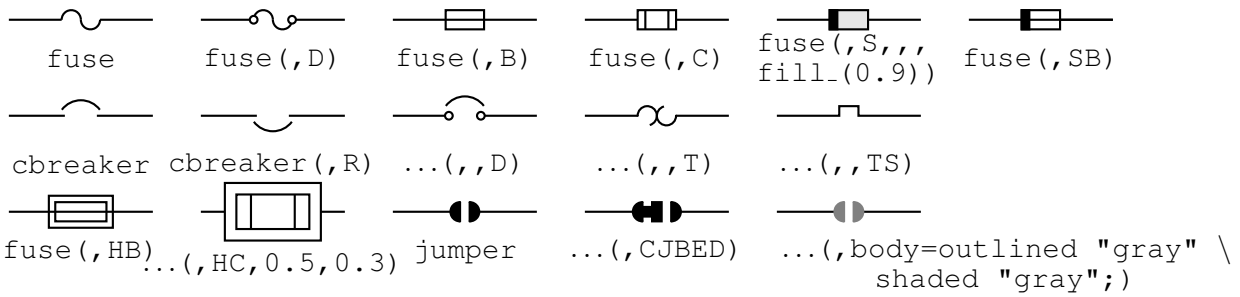


Figure 10: Macros `fuse`, `cbreaker`, and `jumper` `[Fuses.m4]`.

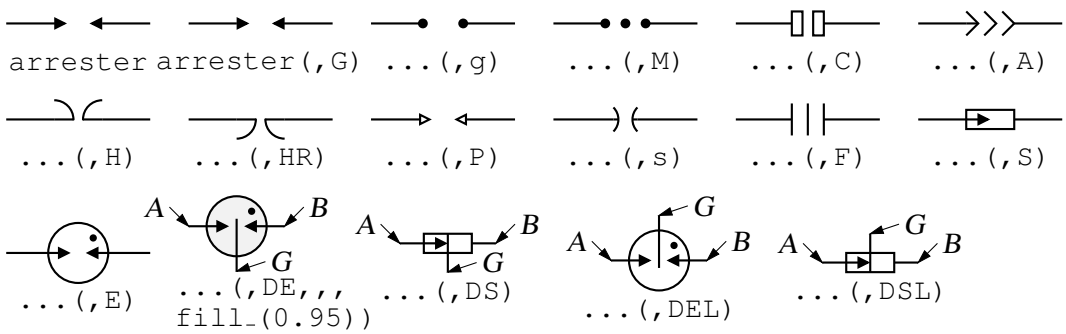


Figure 11: The `arrester` macro `[Arresters.m4]`.

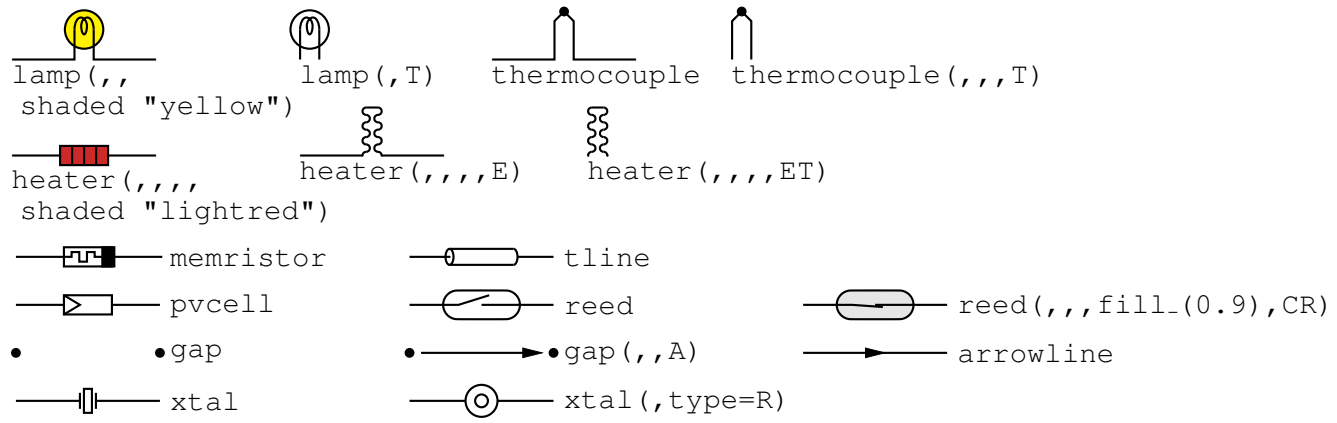


Figure 12: Additional two-terminal elements `[MoreTable.m4]`.

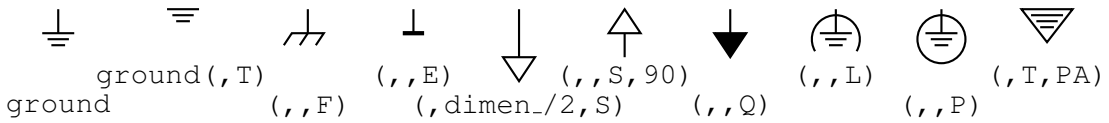


Figure 13: Ground symbols `[Grounds.m4]`.

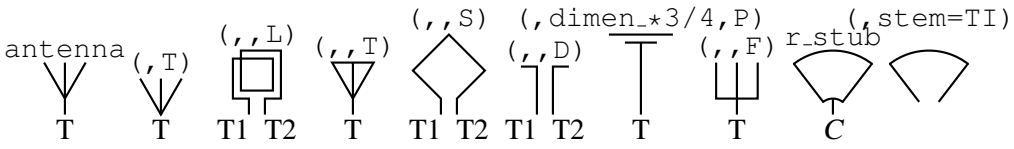


Figure 14: Antenna symbols and the microstrip radial stub `[Antennas.m4]`.

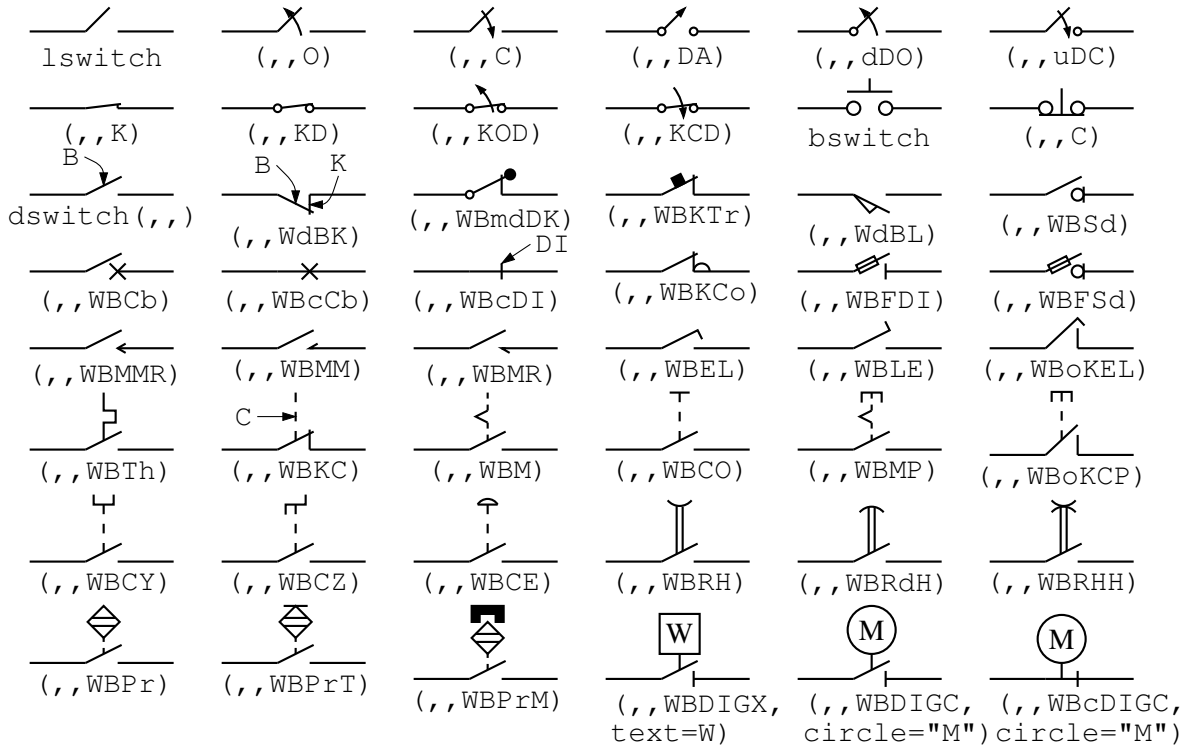


Figure 15: The switch macros; `switch(,,,L|B|D)` is a wrapper for `lswitch`, `bswitch`, and `dswitch` `[Switches.m4]`.

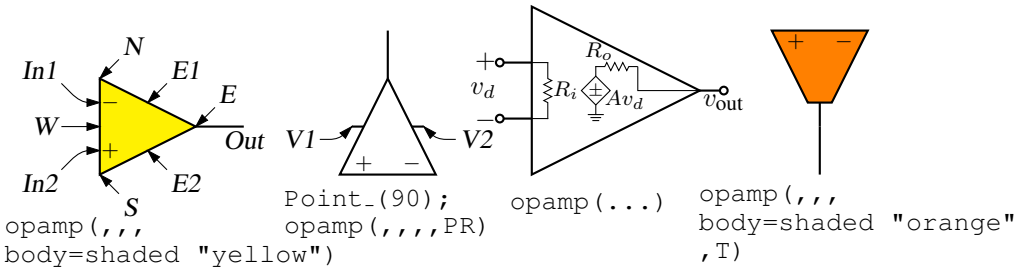


Figure 16: The opamp. The size of the third example is set in the fourth argument and the internal components are defined in the sixth `[Opamp.m4]`.

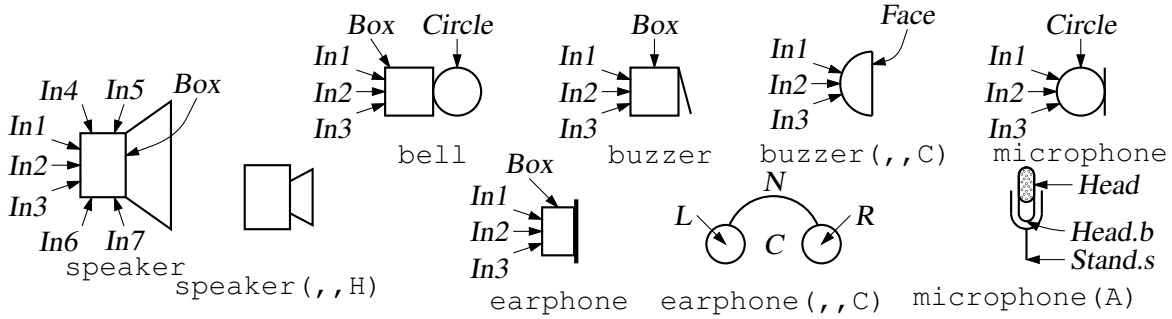


Figure 17: Audio elements `[Audio.m4]`.

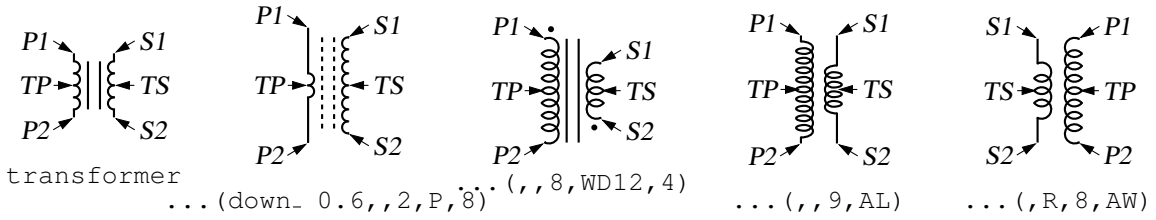


Figure 18: Some variations of the transformer element, drawing direction down [Xform.m4].

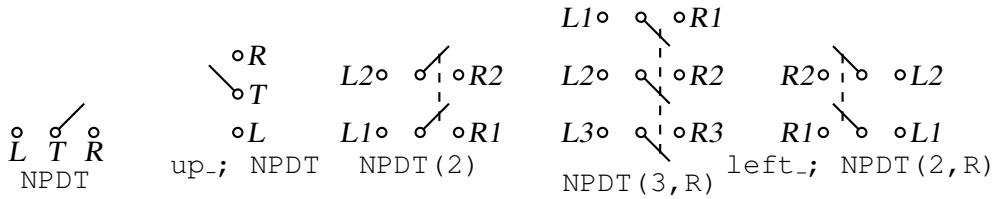


Figure 19: Double throw with the NPDT macro [NPDT.m4].

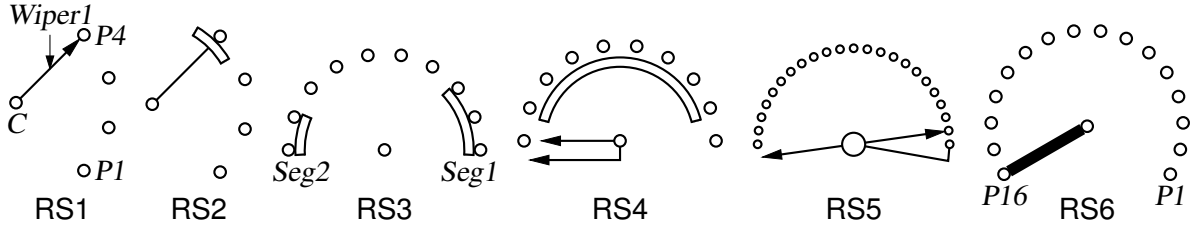


Figure 20: The RotarySwitch macro [RotarySwitch.m4].

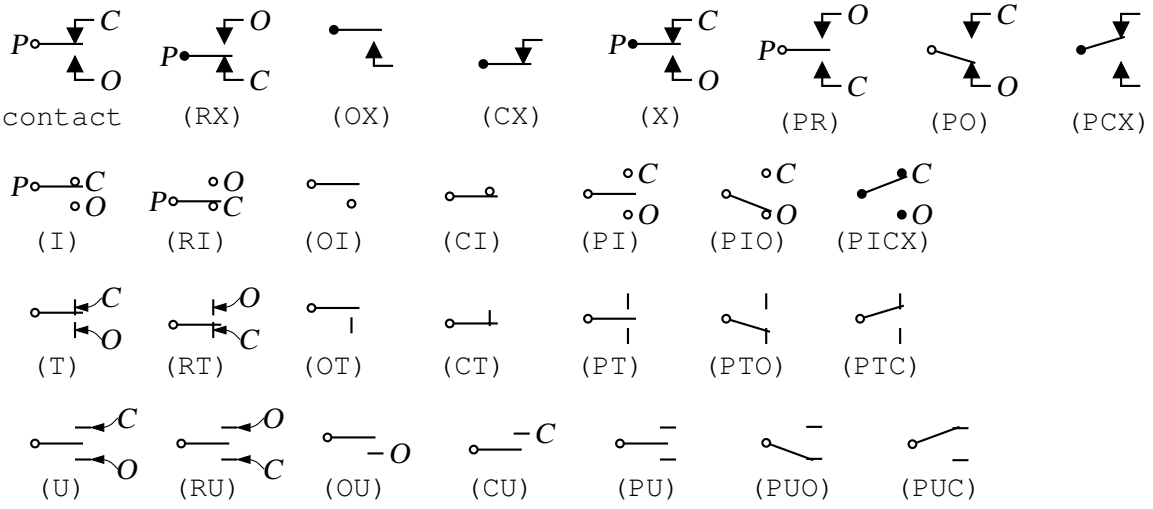


Figure 21: A non-exhaustive sampling of contact macro variations [Contact.m4].

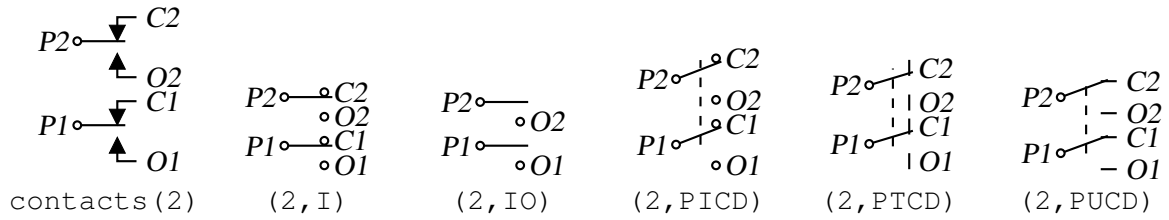


Figure 22: The `contacts` macro `[Contacts.m4]`.

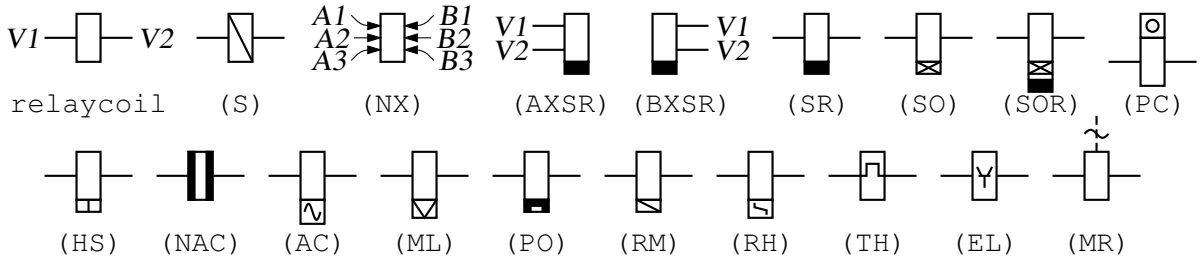


Figure 23: The `relaycoil` macro `[relaycoil.m4]`.

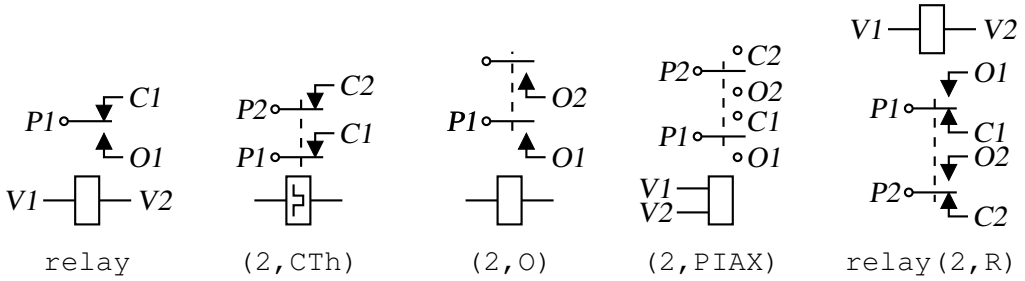


Figure 24: Some variants of `relay` `[Relay.m4]`.

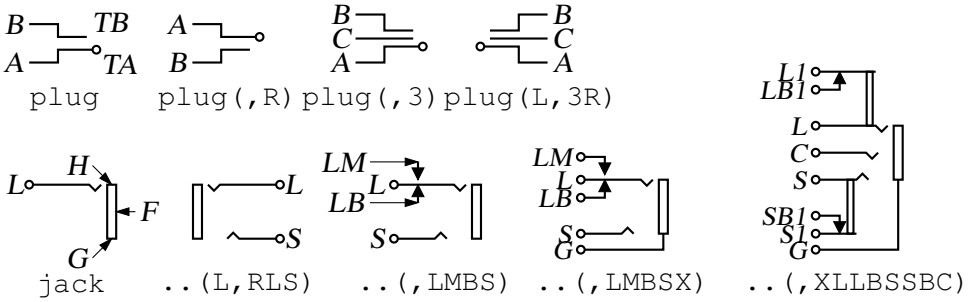


Figure 25: The `jack` and `plug` macros `[Jack.m4]`.

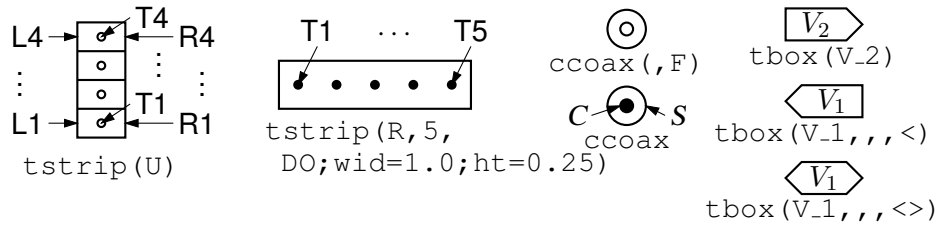


Figure 26: The `tstrip`, `ccoax`, and `tbox` macros `[Tstrip.m4]`.

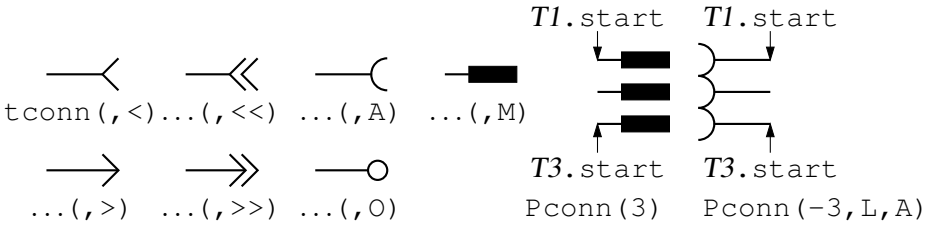


Figure 27: The `tconn` and `Pconn` macros `[Conn.m4]`.

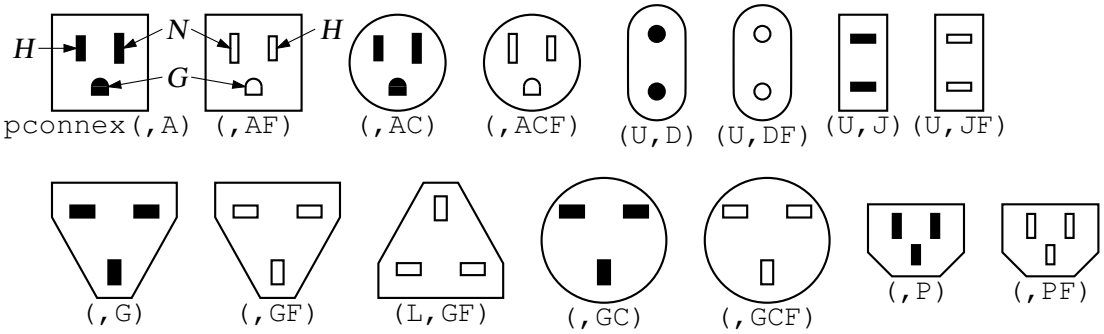


Figure 28: The `pconnex` macro `[Pconn.m4]`.

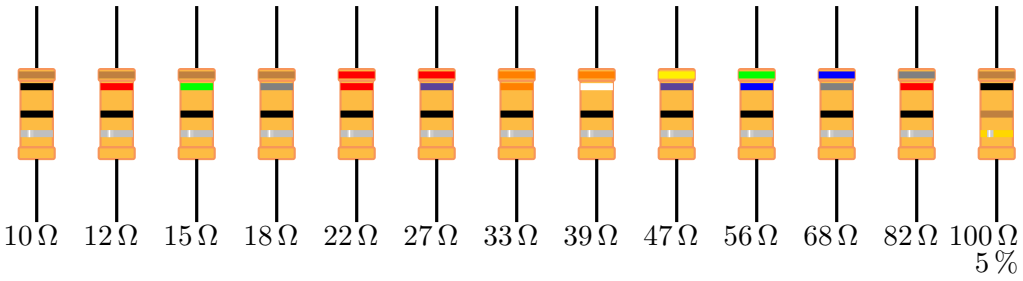


Figure 29: Color-coded through-hole resistors. All are 10 percent tolerance except the last, which illustrates 5 percent `[cbresistor.m4]`.

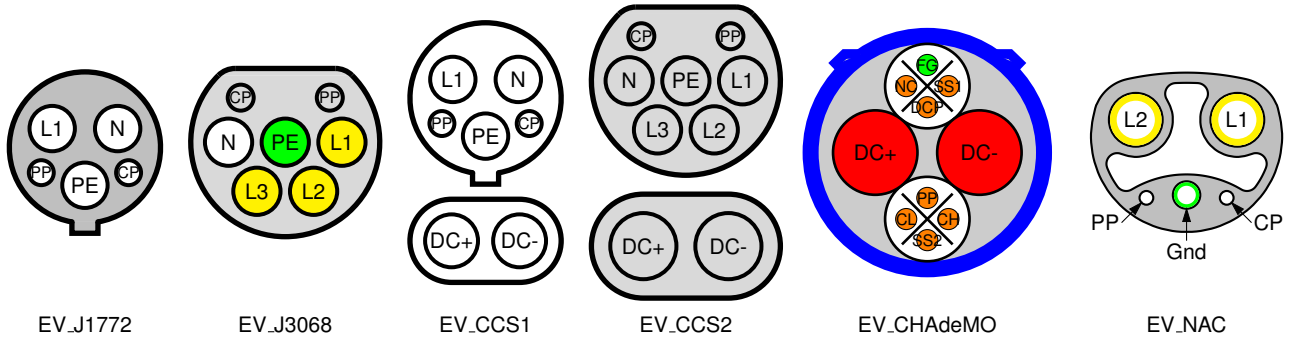


Figure 30: Electric vehicle charging plug patterns make extensive use of *key=value* pairs to set options `[EVplugs.m4]`.

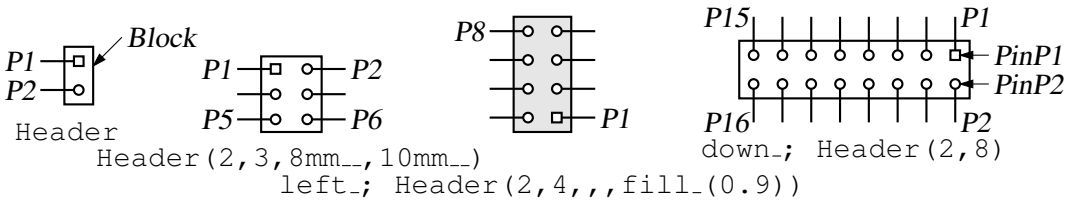


Figure 31: The `Header` macro `[Headers.m4]`.

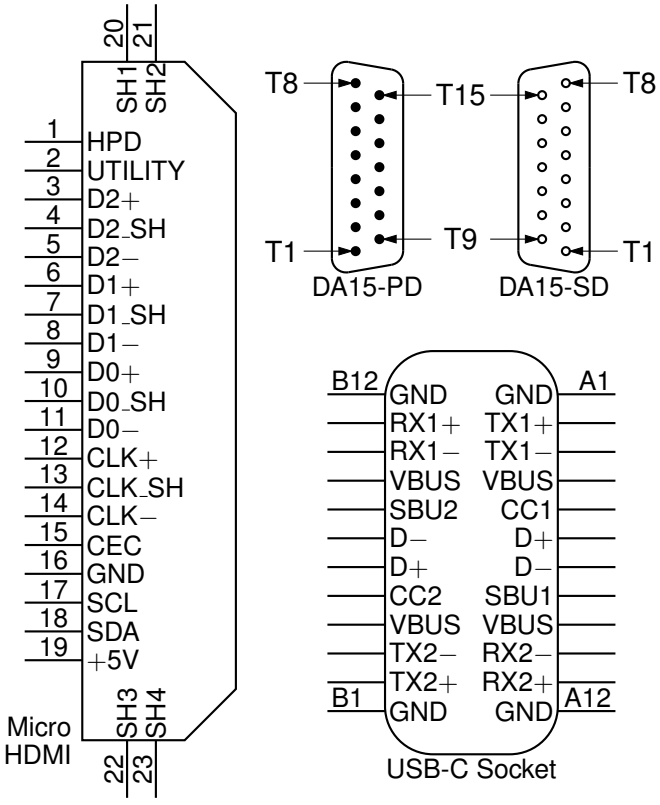


Figure 32: Some connectors with simple geometry and lists of labels `[Sockets.m4]`.

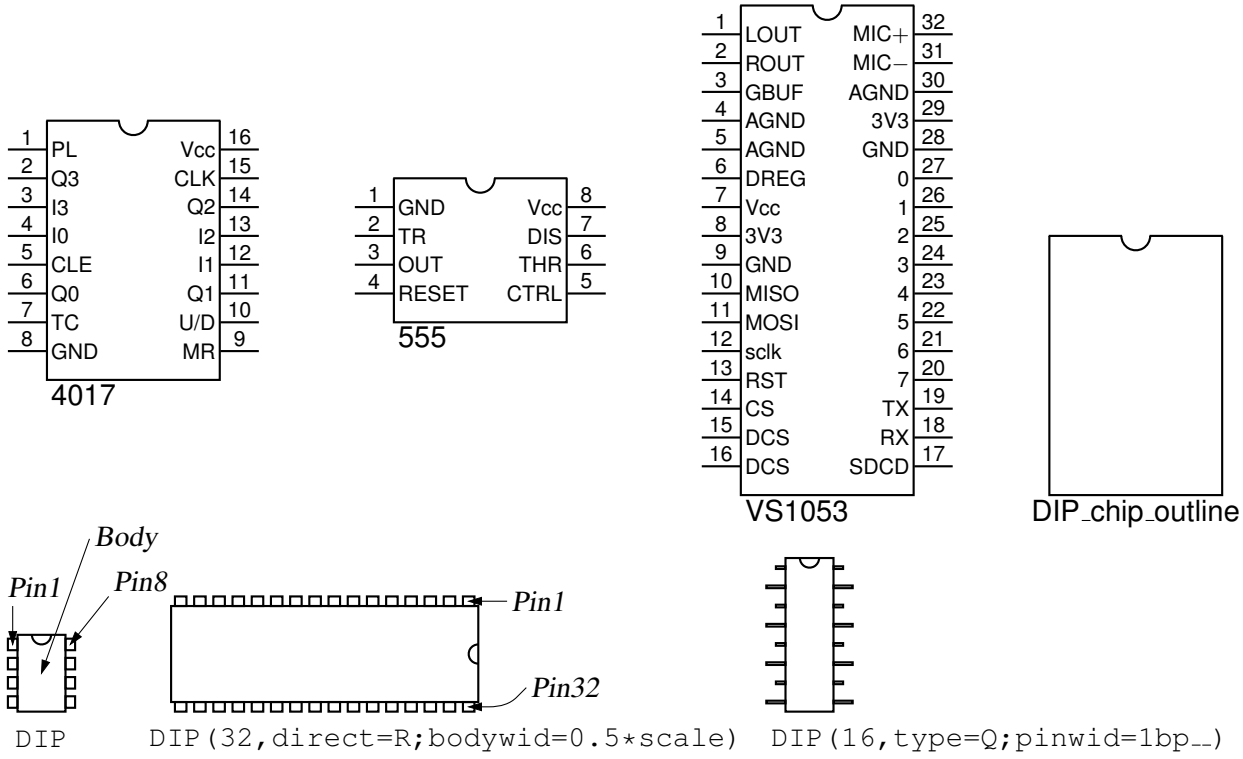


Figure 33: IC package outlines and examples [Chips.m4].

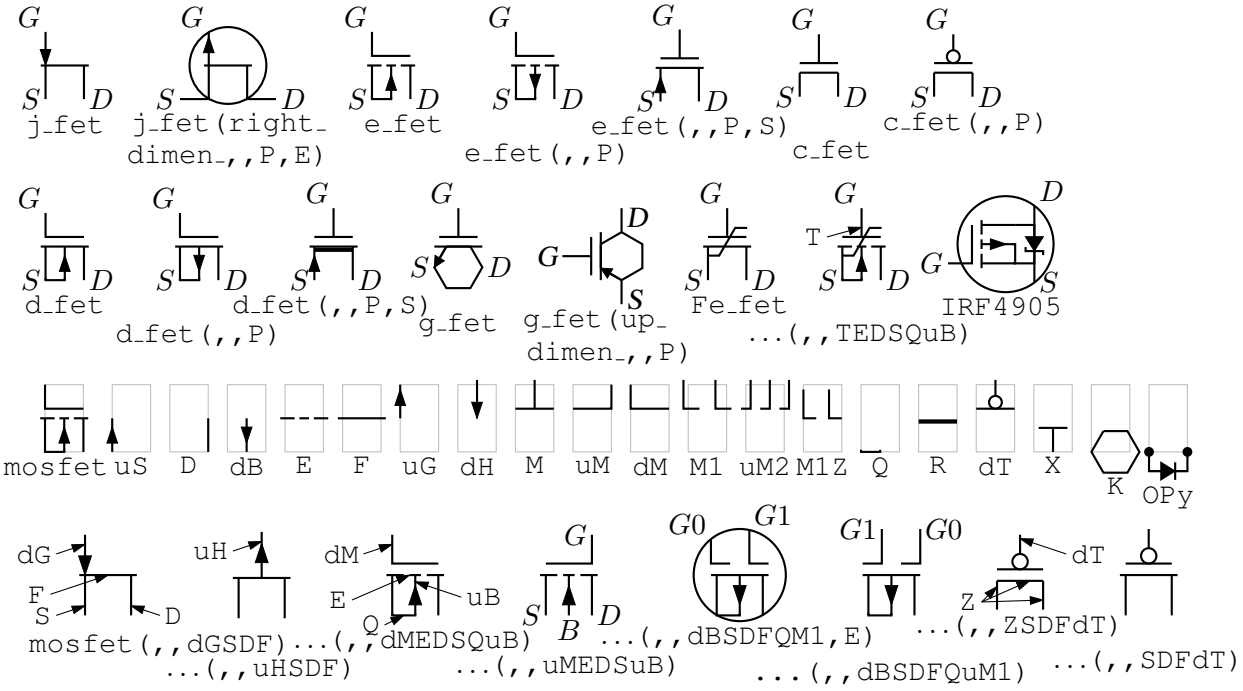


Figure 34: FETs, showing programmable components and example customizations [fet.m4].

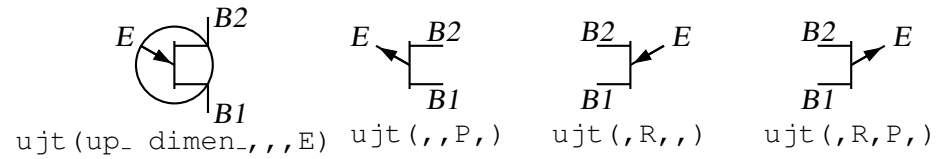


Figure 35: UJT examples [ujt.m4].

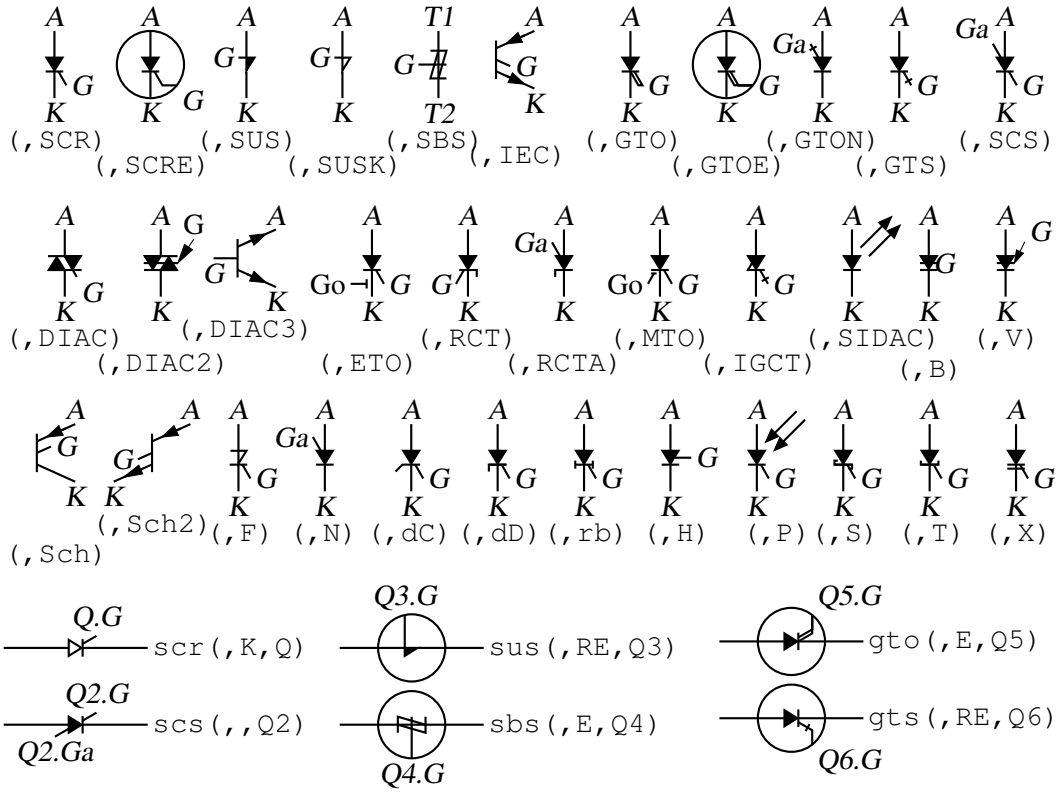


Figure 36: Thyristor examples. The thyristor is a 3- or 4-terminal composite element [Thyristor.m4].

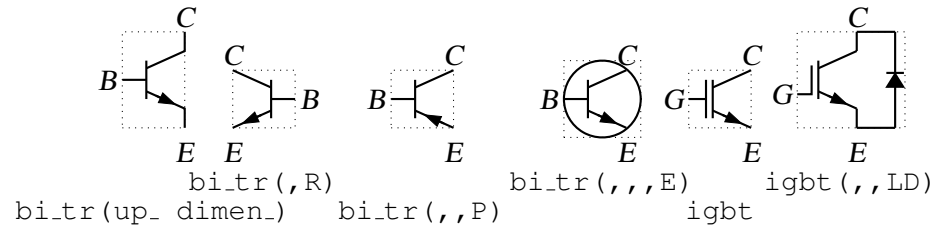


Figure 37: Bipolar transistors (drawing direction: up) [Bip.m4].

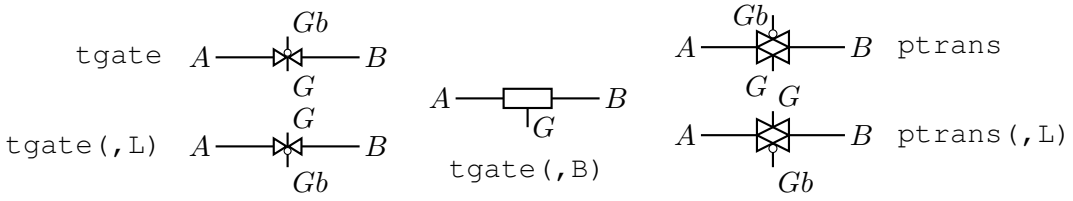


Figure 38: The `tgate` and `ptrans` elements `[Tgate.m4]`.

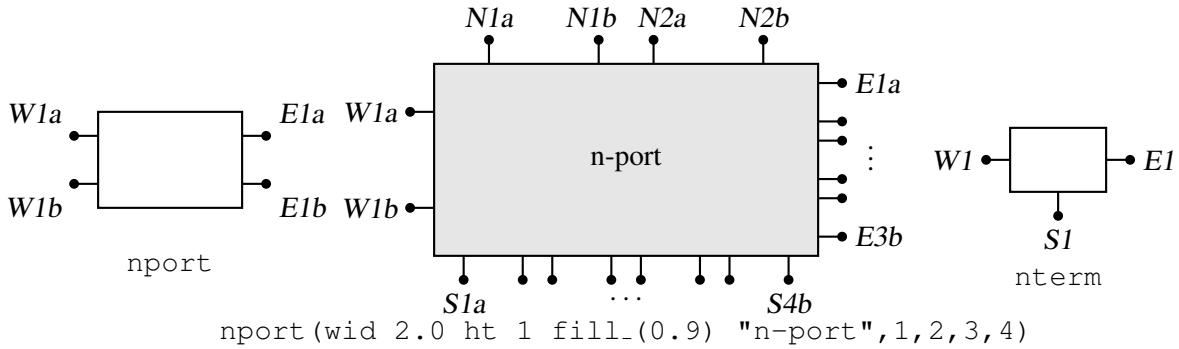


Figure 39: The `nport` and `nterm` macros `[Nport.m4]`.

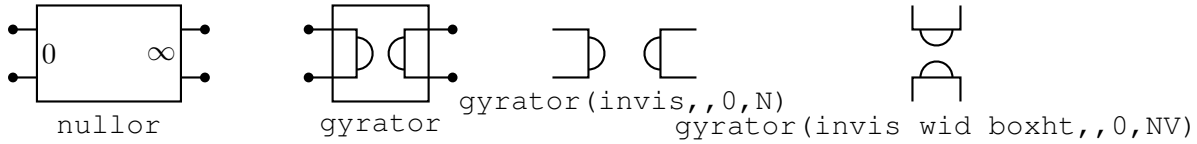


Figure 40: Some customizations of `nport [NLG.m4]`.

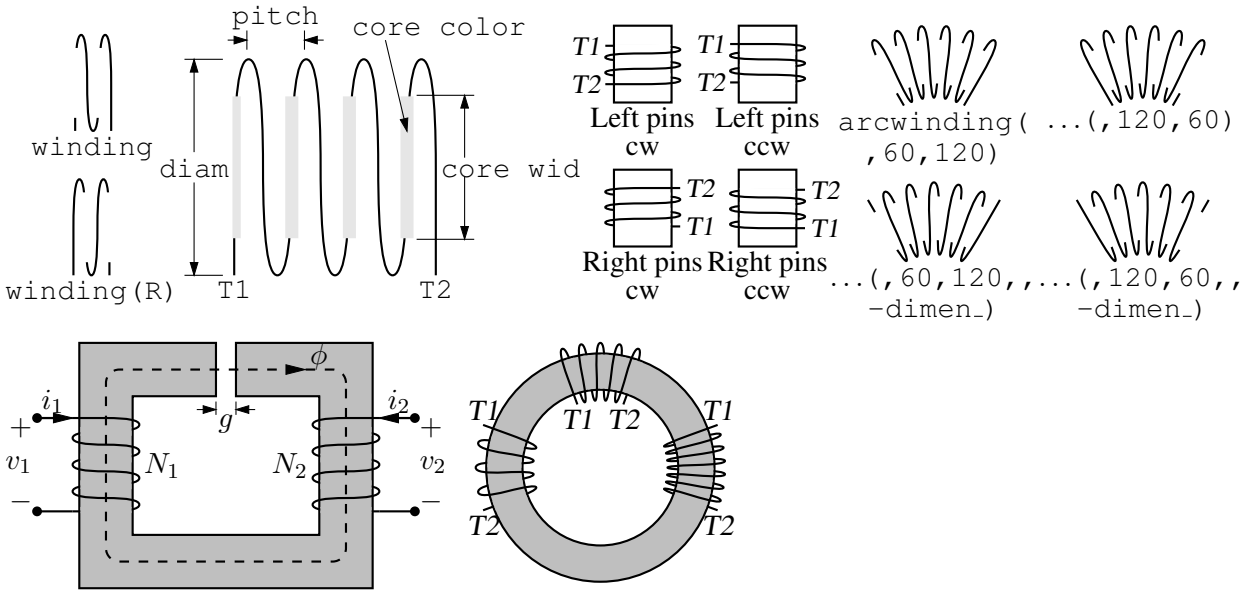


Figure 41: The macros `winding(L|R,winding diam,pitch,nturns,core wid, "core color")` and `arcwinding(winding diam, start deg, enddeg, nturns, core centre rad, core wid, "core color")` [Windings.m4].

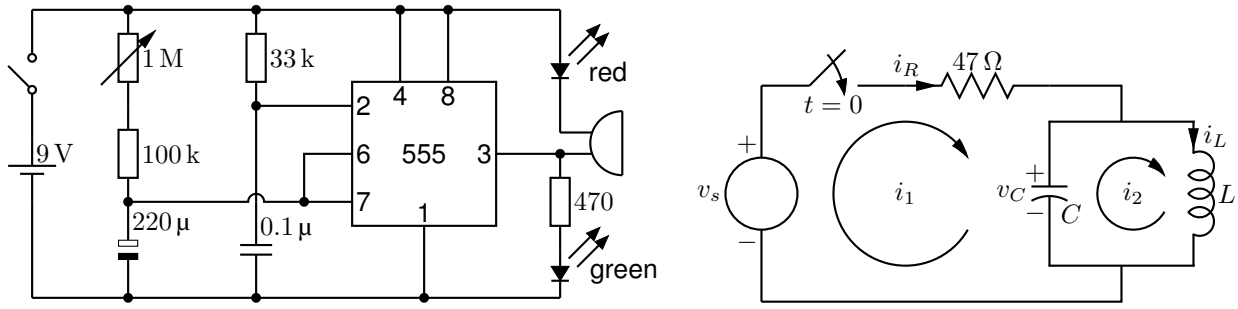


Figure 42: Two simple labeled circuits [ex01.m4].

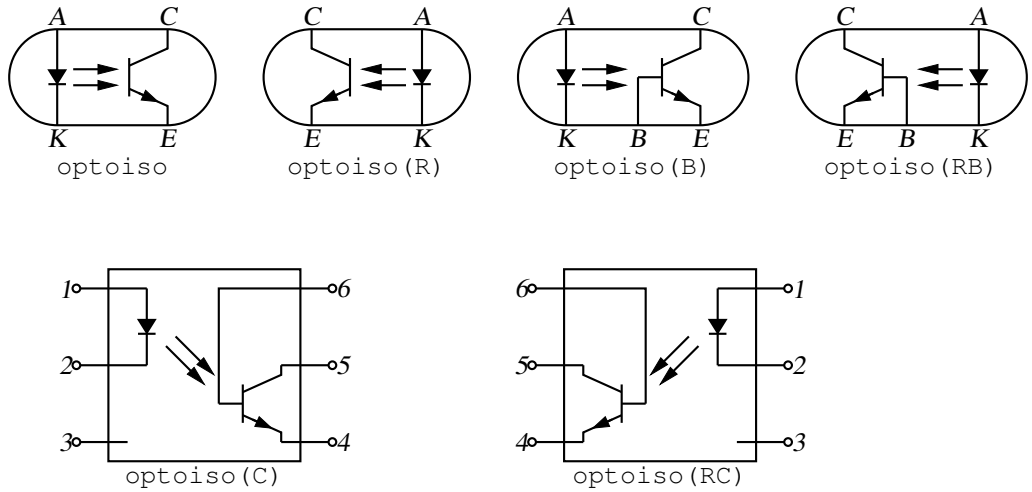


Figure 43: Optical isolator: a circuit with right or left orientation [Optoiso.m4].

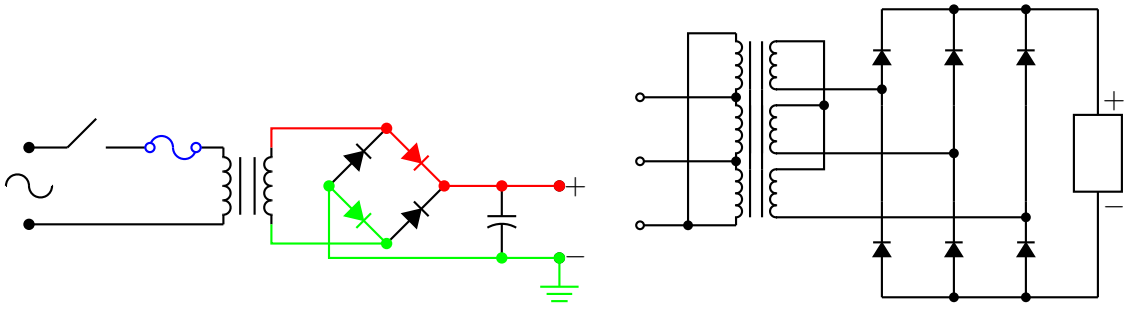


Figure 44: An elementary power supply circuit with colored elements, and a multiple-winding transformer with 3-phase rectifier `[pwrsupply.m4]`.

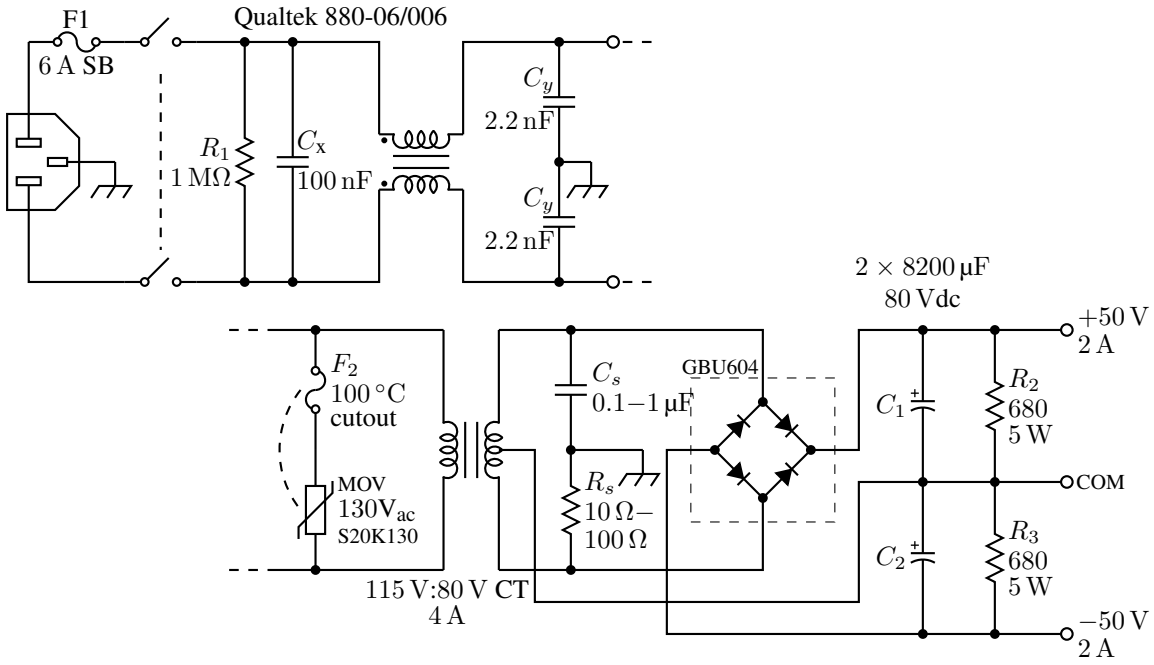


Figure 45: An unregulated 50V power supply `[PS50.m4]`.

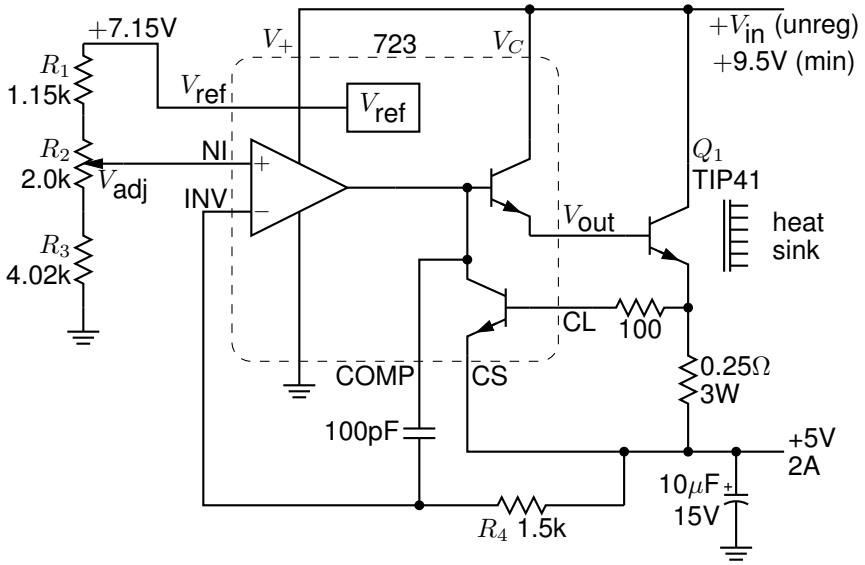


Figure 46: A 723 regulator circuit with a heat sink `[Reg723.m4]`.

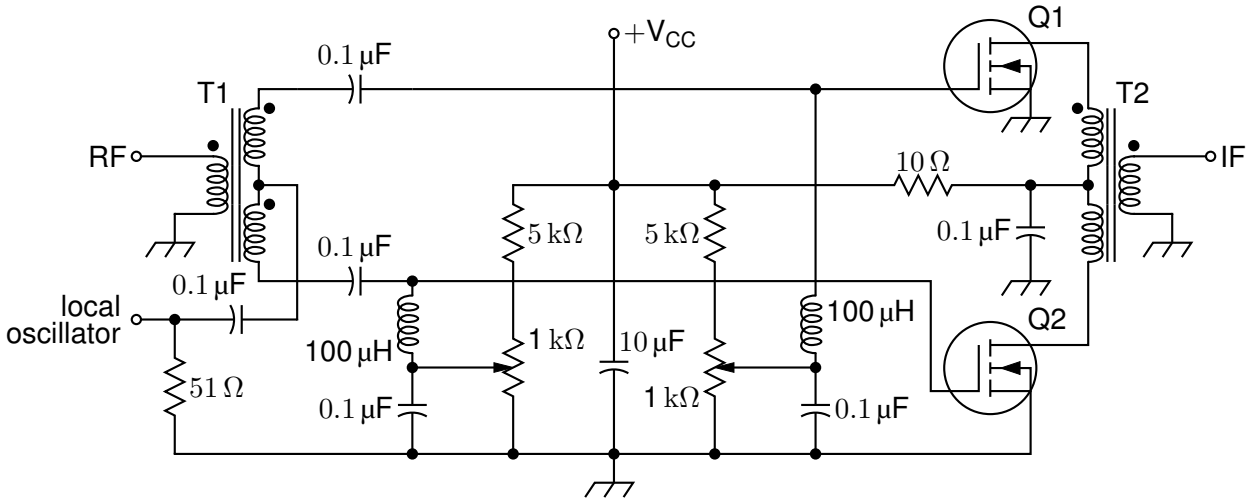


Figure 47: A balanced mixer, using `mosfet` and a custom transformer `[Mixer.m4]`.

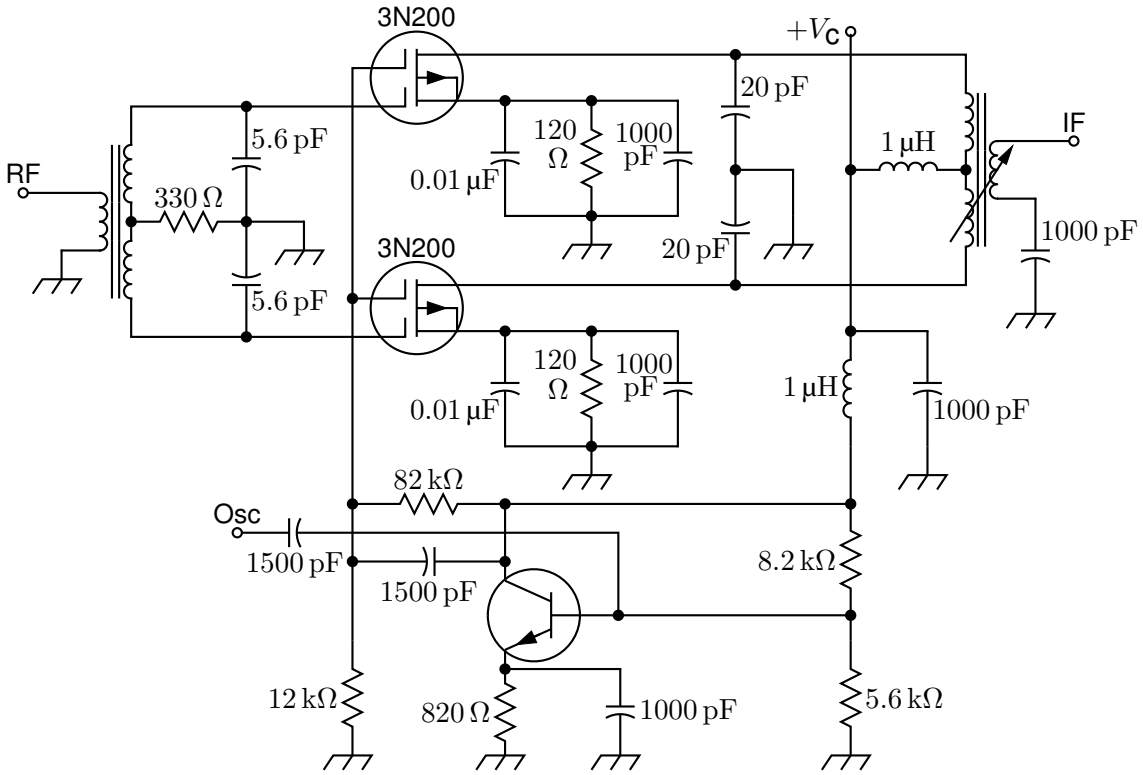


Figure 48: A push-pull mixer, showing FETs with multiple gates `[PushPull.m4]`.

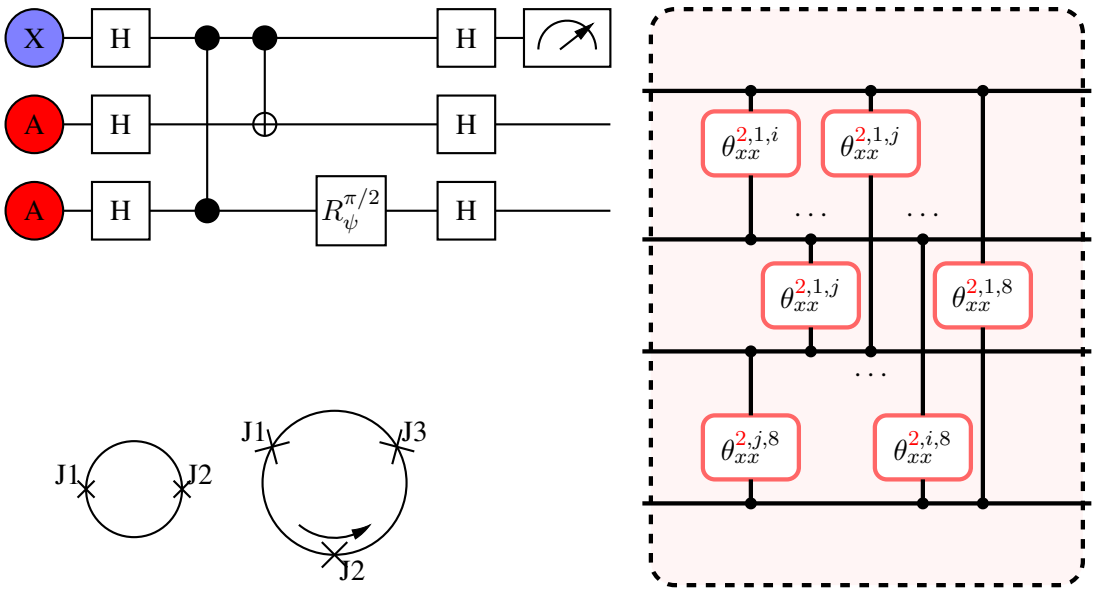


Figure 49: Quantum circuits `[Quantum.m4]`.

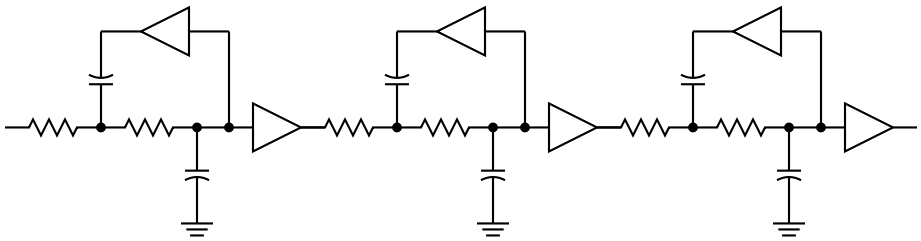


Figure 50: A six-pole filter `[Sixpole.m4]`.

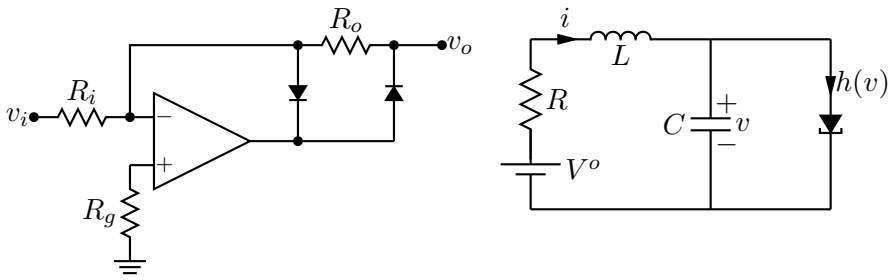


Figure 51: Precision half-wave rectifier and a tunnel diode circuit (illustrating `opamp`, `diode`, `resistor`, `ground`, and labels) `[ex18.m4]`.

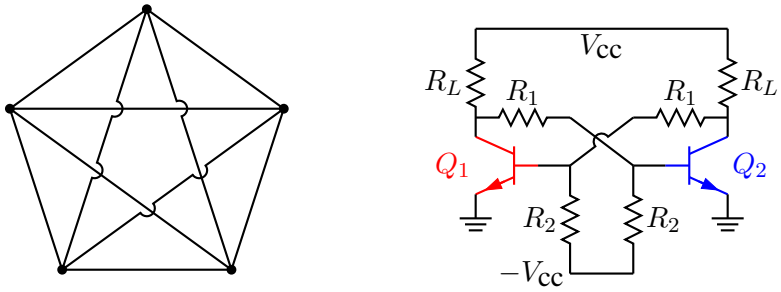


Figure 52: Non-planar graph and bistable circuit (illustrating the `crossover` macro and colored elements) `[ex10.m4]`.

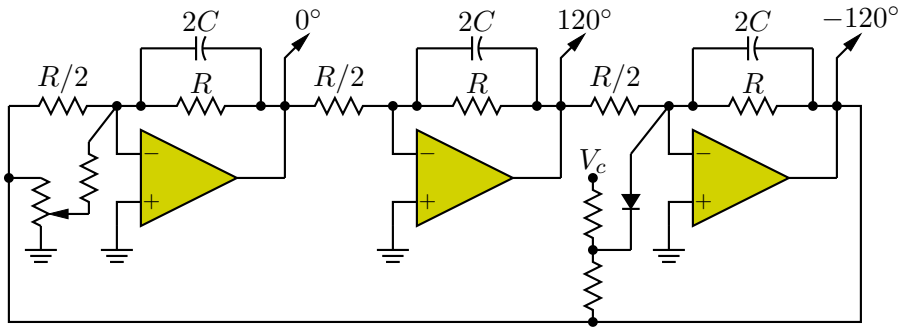


Figure 53: Three-phase oscillator `[Three.m4]`.

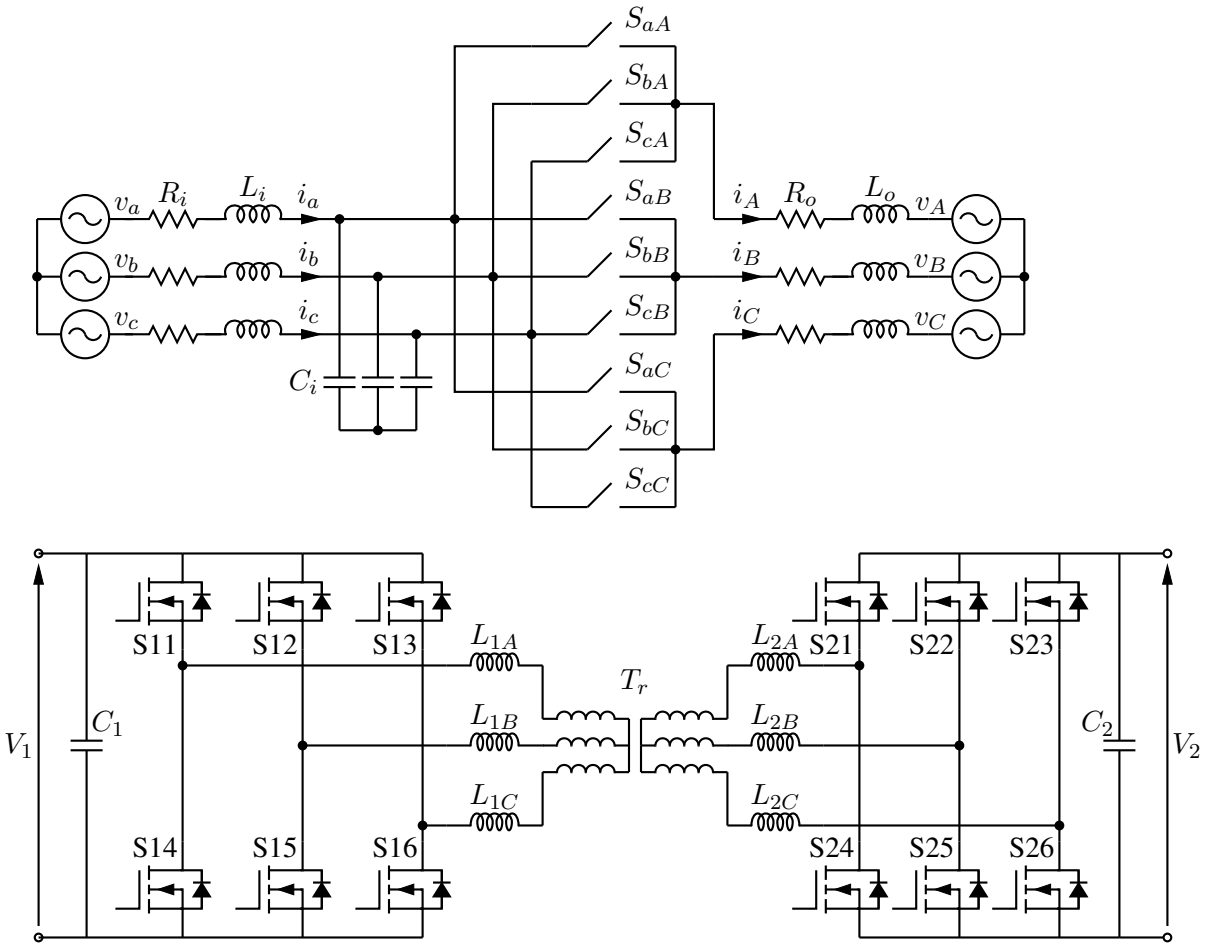


Figure 54: A three-phase switched AC-AC converter and a DC-DC converter `[MC.m4]`.

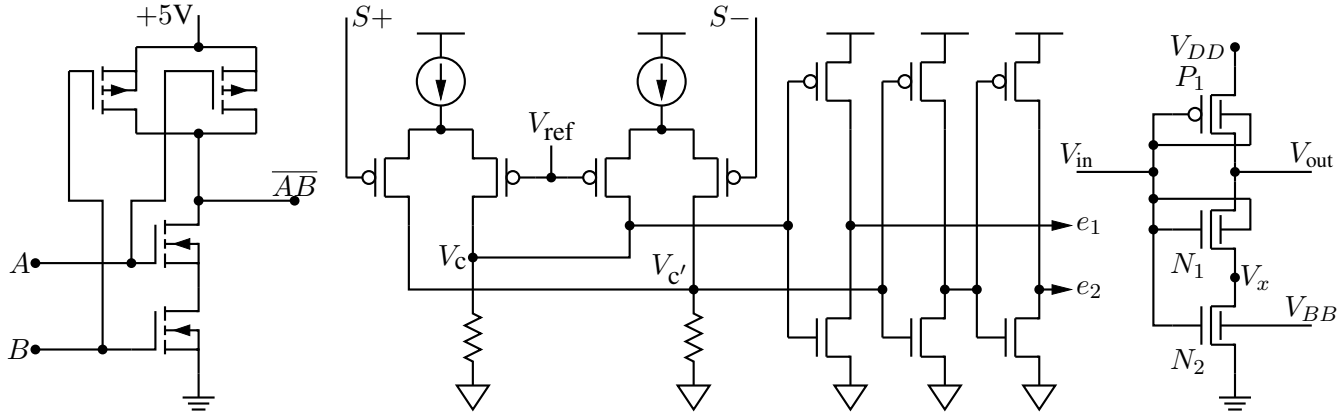


Figure 55: A CMOS NAND gate, a test circuit, and an XMOSFET example `[ex12.m4]`.

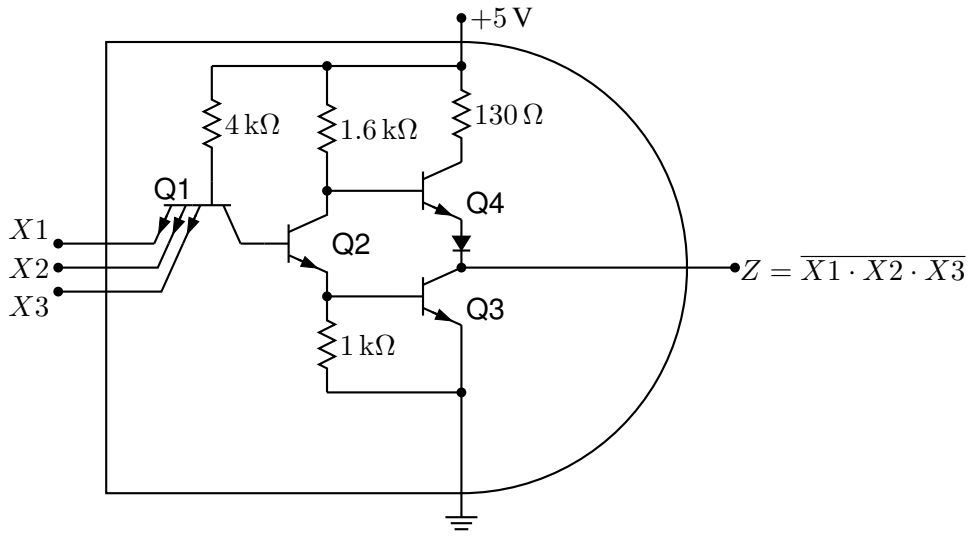


Figure 56: TTL NAND gate illustrating a transistor with multiple emitters `[TTLnand.m4]`.

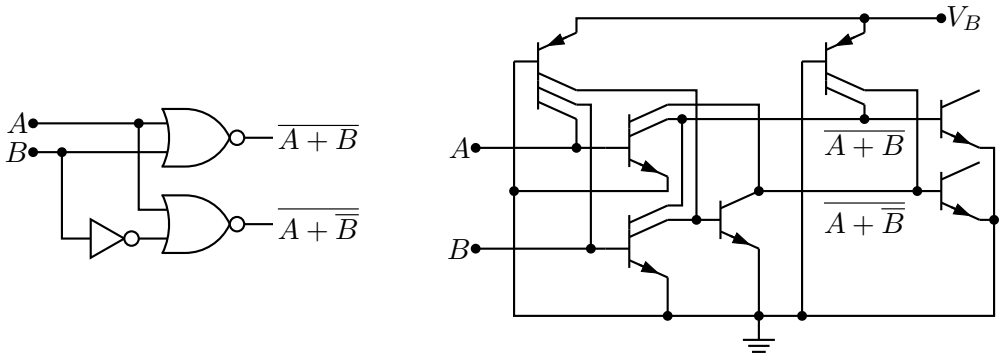


Figure 57: Gate circuit and equivalent embedded I^2L components illustrating multiple collectors `[I2L.m4]`.

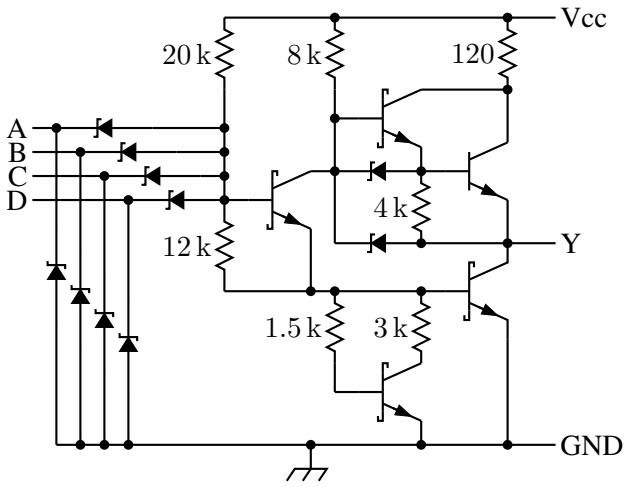


Figure 58: A 4-input NAND circuit illustrating the `S` (Schottky) option of `bi_trans [Schottky.m4]`.

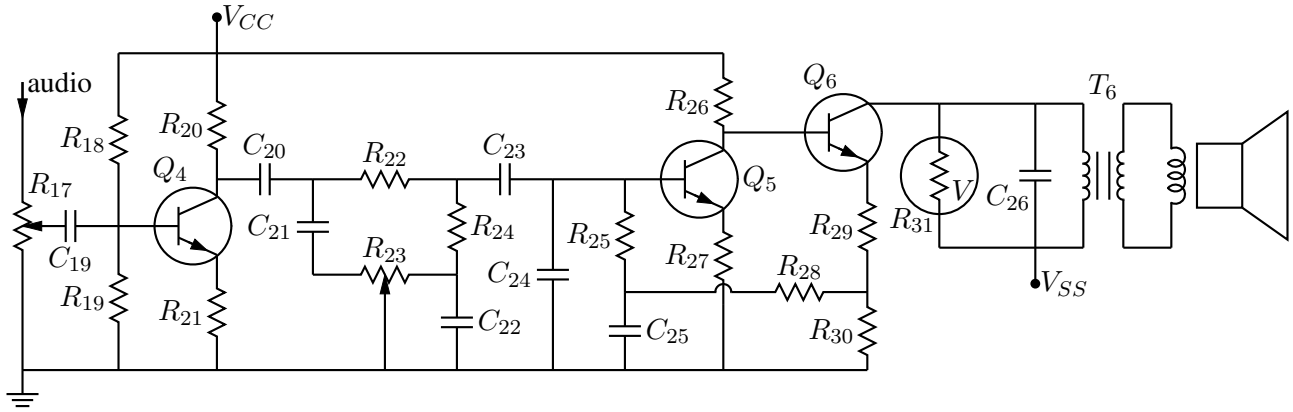


Figure 59: Transistor radio audio chain [ex11.m4].

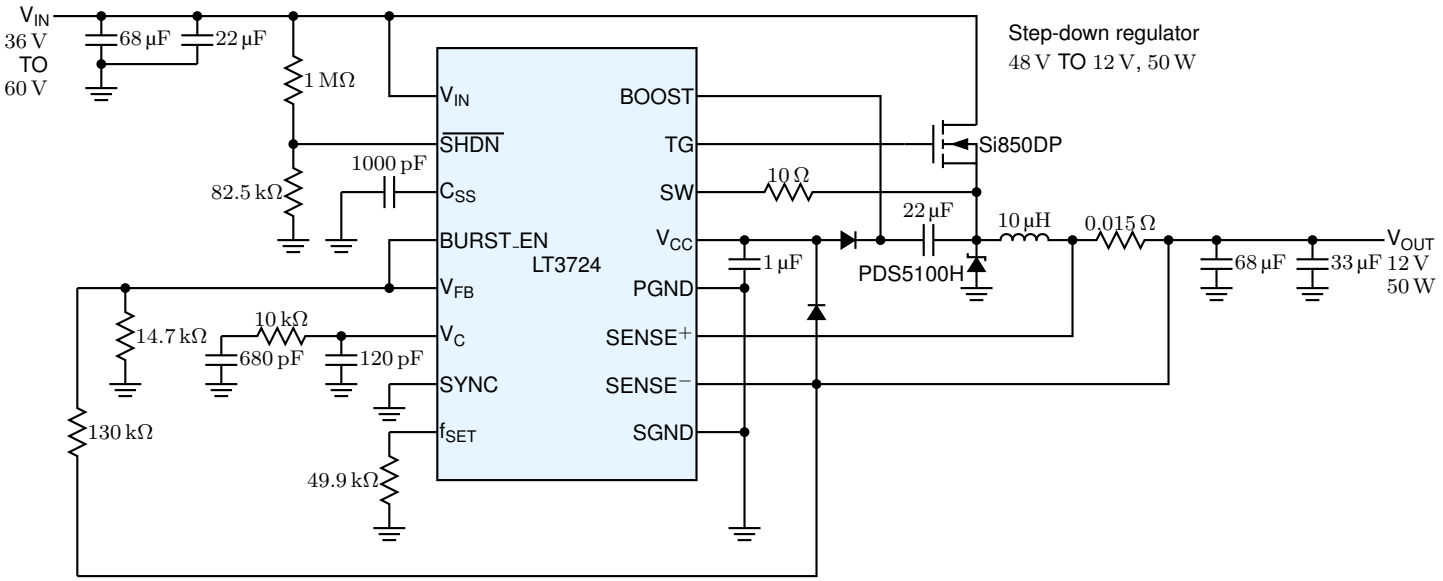


Figure 60: IC controller and auxiliary elements of a step-down converter [LT3524.m4].

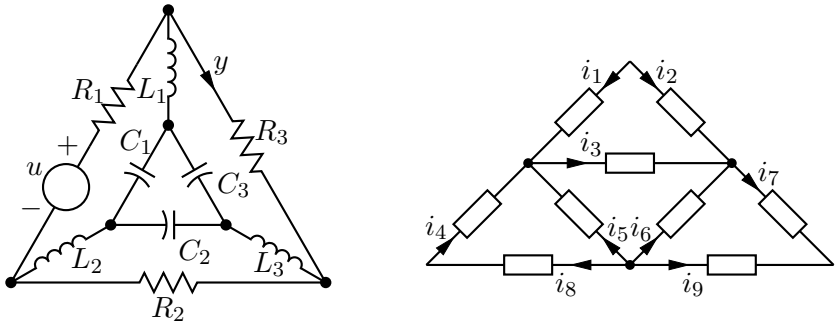


Figure 61: Labels on non-manhattan elements [ex04.m4].

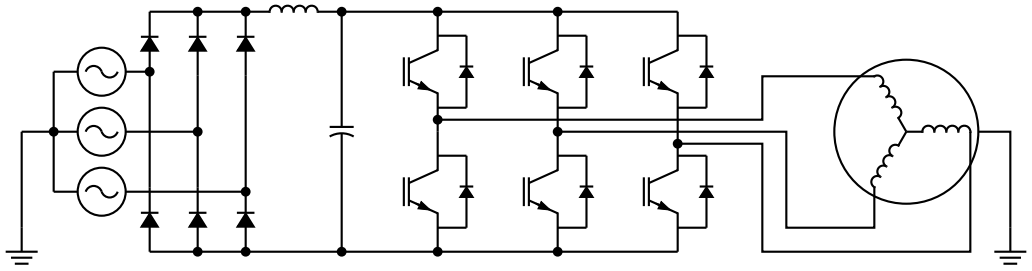


Figure 62: Synchronous machine driven by variable-speed drive and rectifier [Drive.m4].

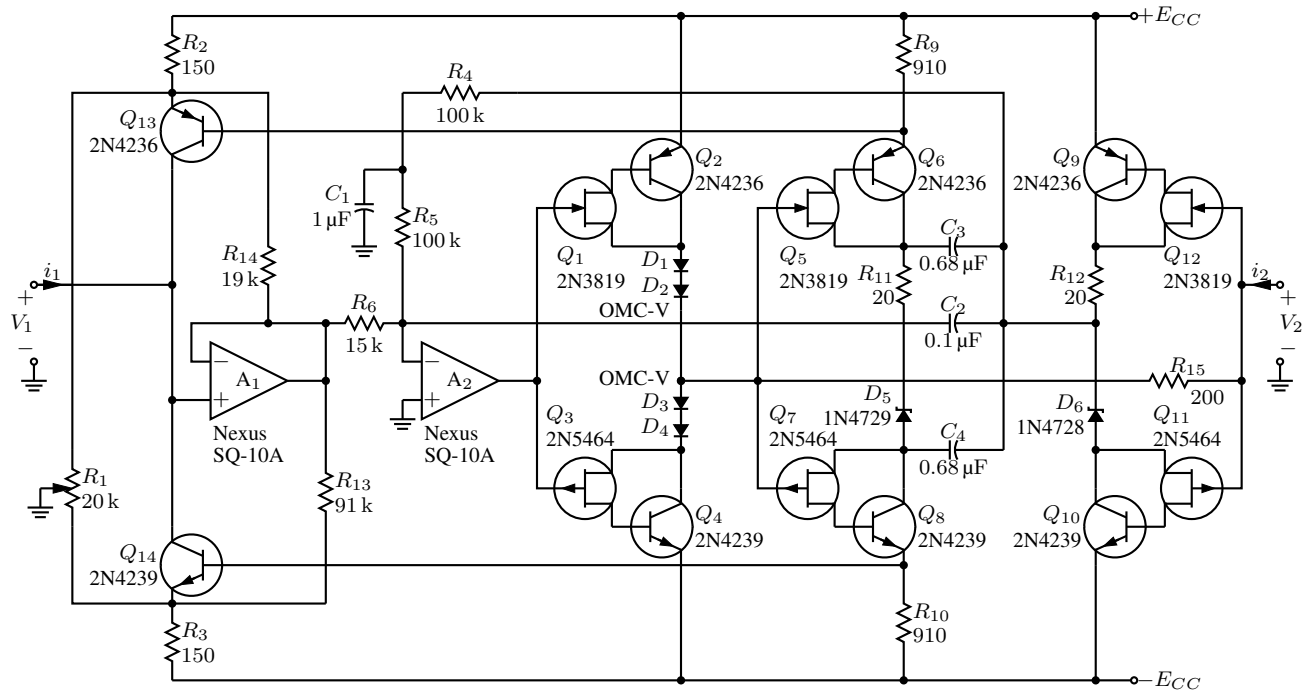


Figure 63: Realization of a controlled source (illustrating stacked element labels) [Csource.m4].

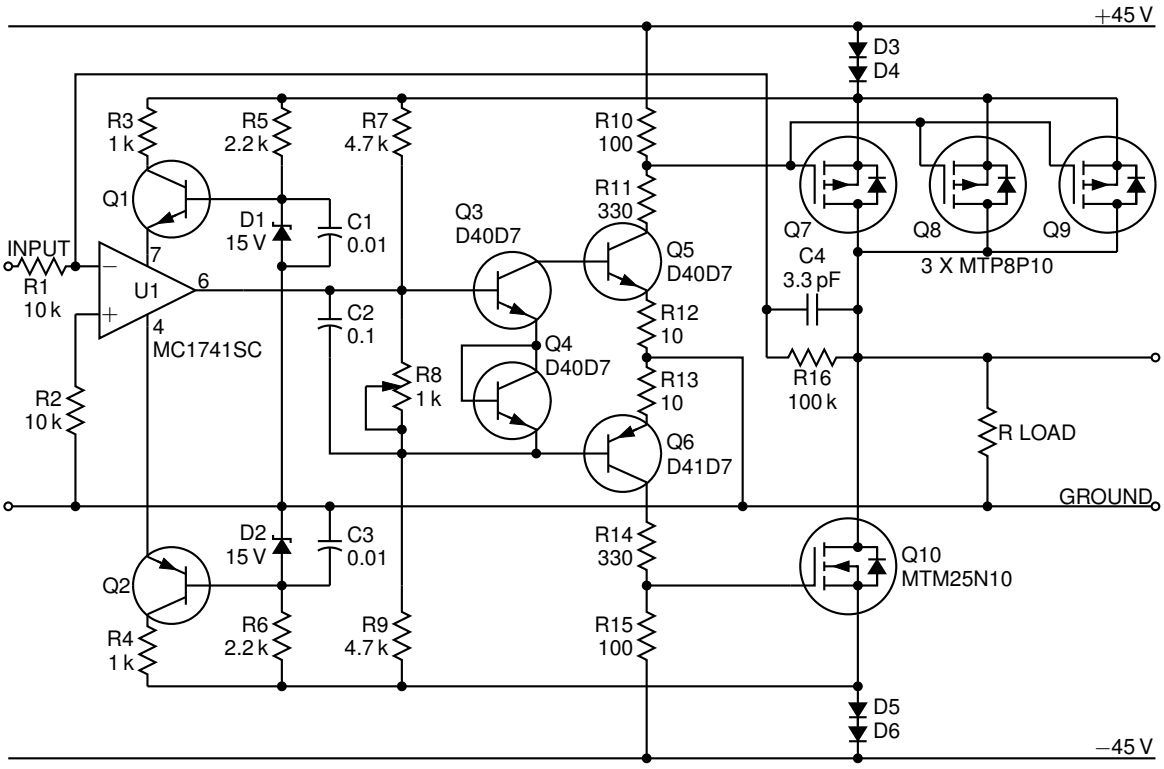


Figure 64: A 50 W audio amplifier [AudioAmp.m4].

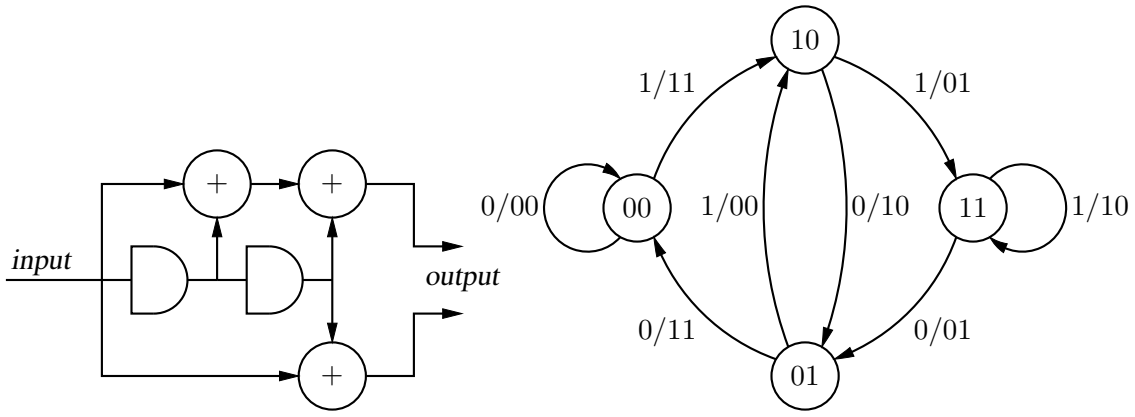


Figure 65: A rate $1/2$ binary convolutional coder and its state diagram [ex16.m4].

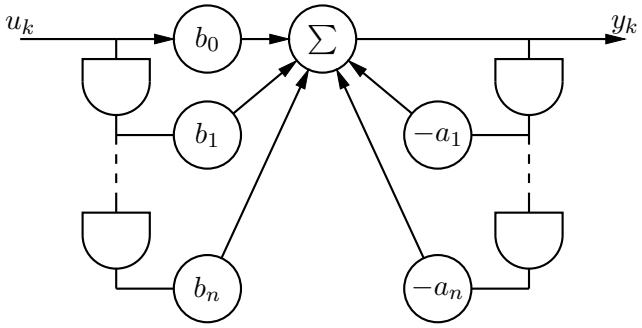


Figure 66: Digital filter `[ex03.m4]`.

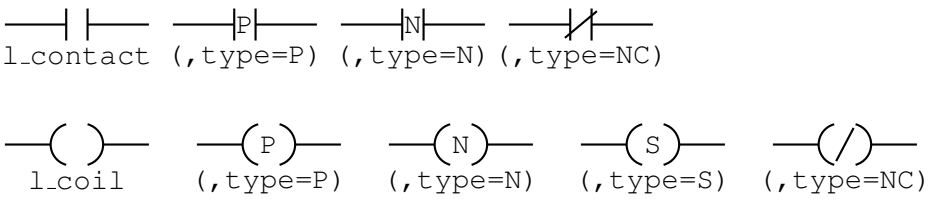


Figure 67: Ladder-diagram symbols `[Ladder.m4]`.

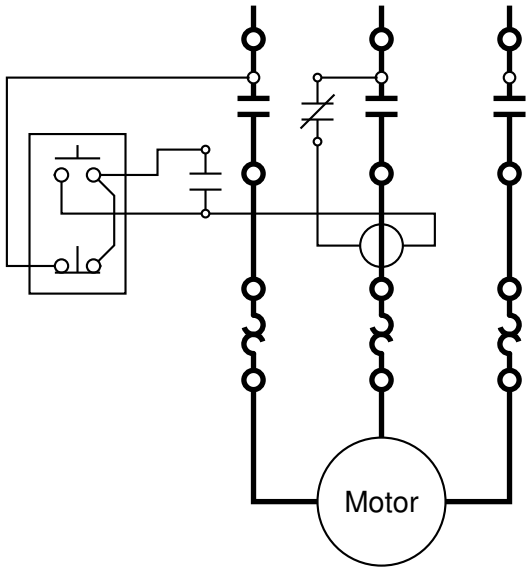


Figure 68: Motor control connections `[MotorControl.m4]`.

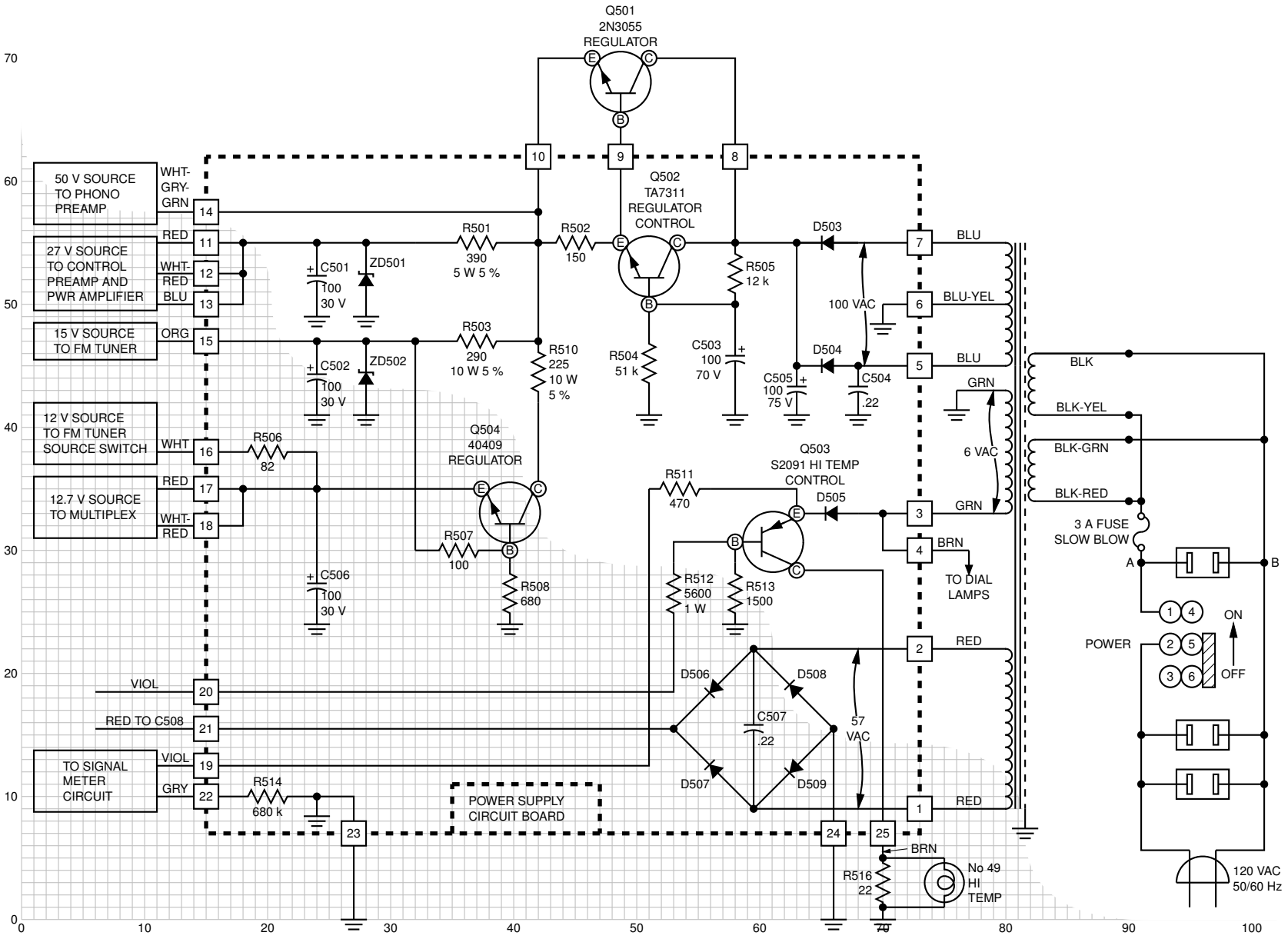


Figure 69: The power supply of a Heathkit AR-15 (Now, *that* was a receiver!) with custom transformer and other elements, drawn on a grid (partially shown) to aid in placement `[Heathkit.m4]`.

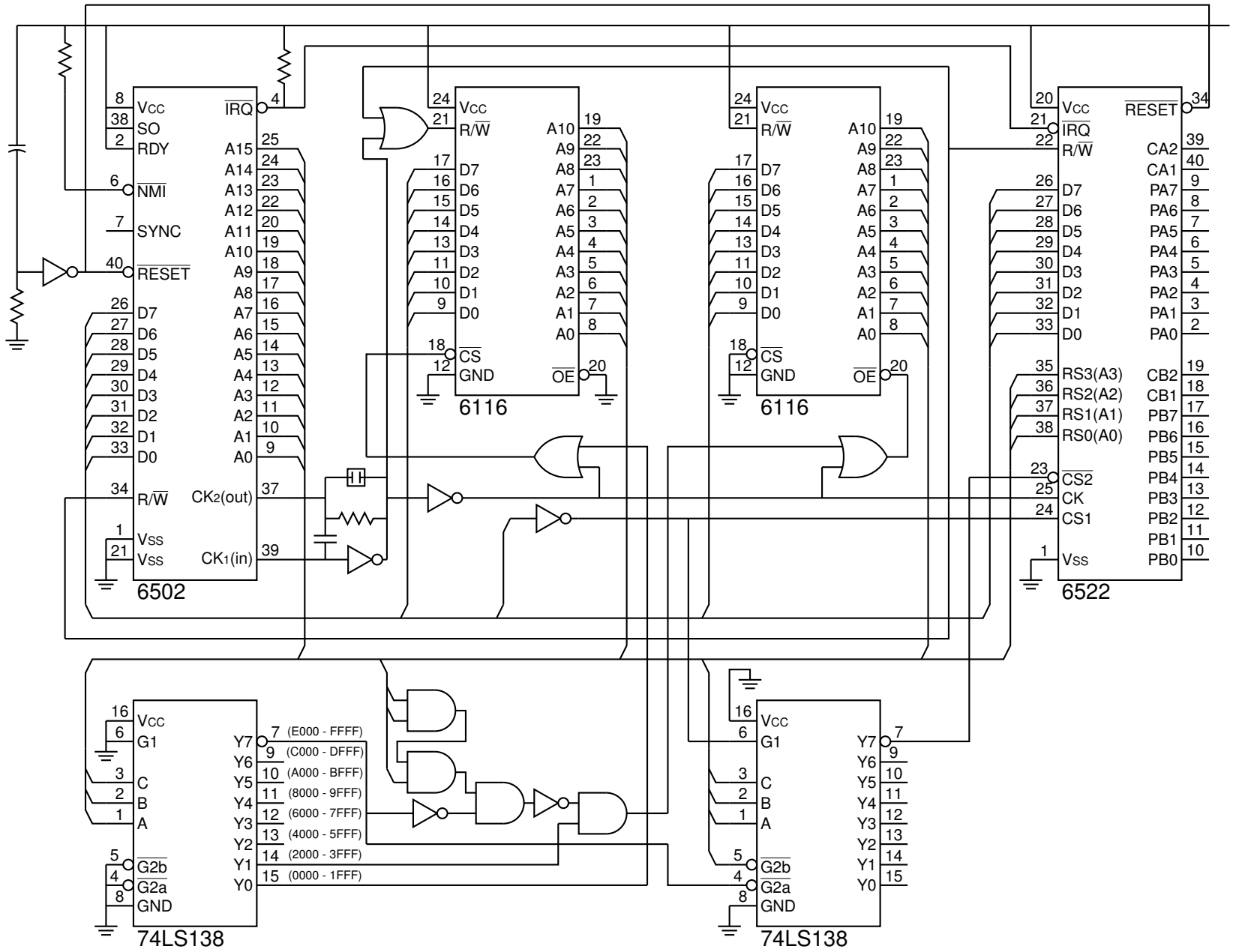


Figure 70: A digital circuit of moderate size, redrawn from M. P. Maclenan and G. M. Burns, “An Approach to Drawing Circuit Diagrams for Text Books,” Tugboat (12)1, March 1991, pp. 66-69 `[lcct.m4]`.

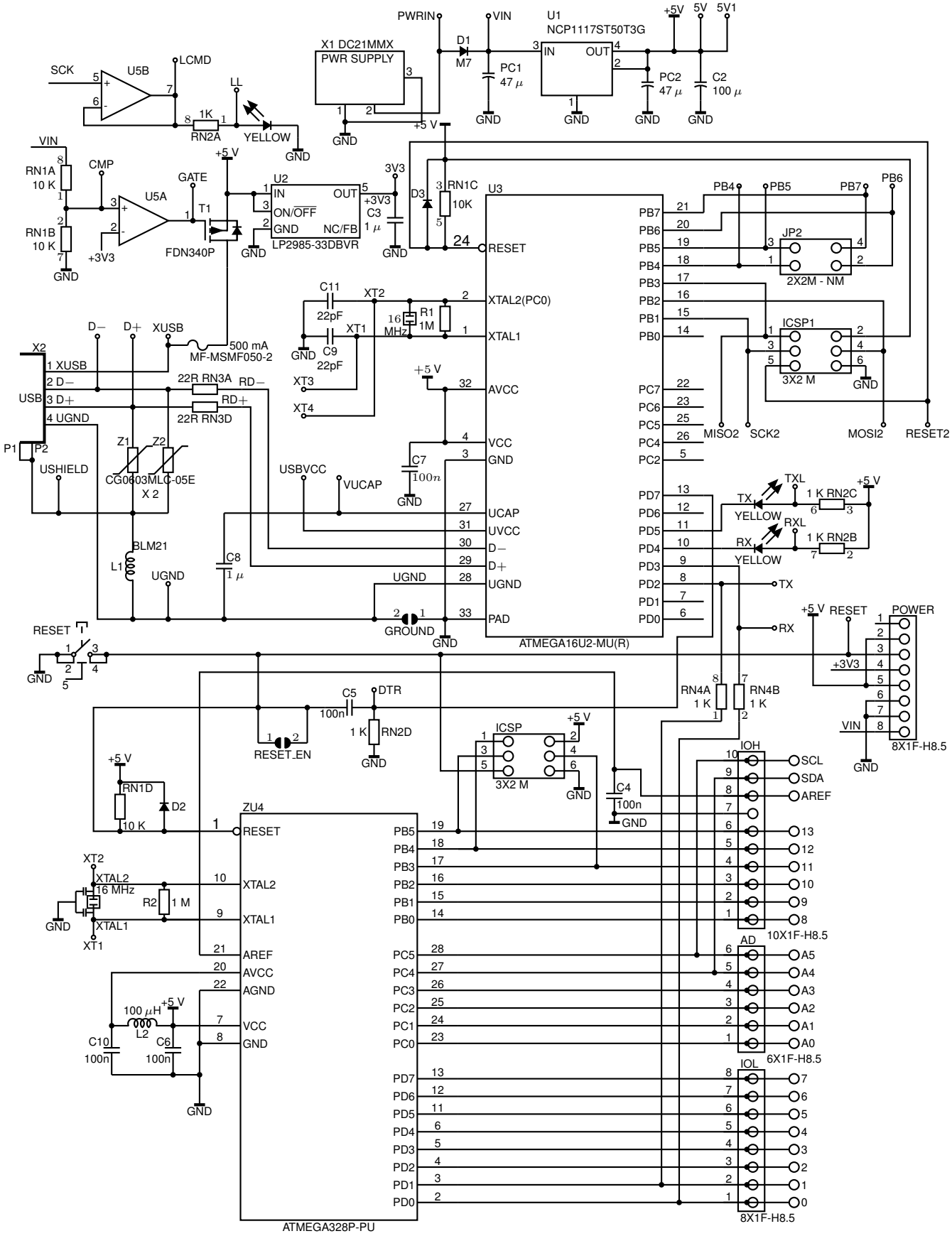


Figure 71: An Arduino UNO circuit adapted and redrawn [UNO.m4].

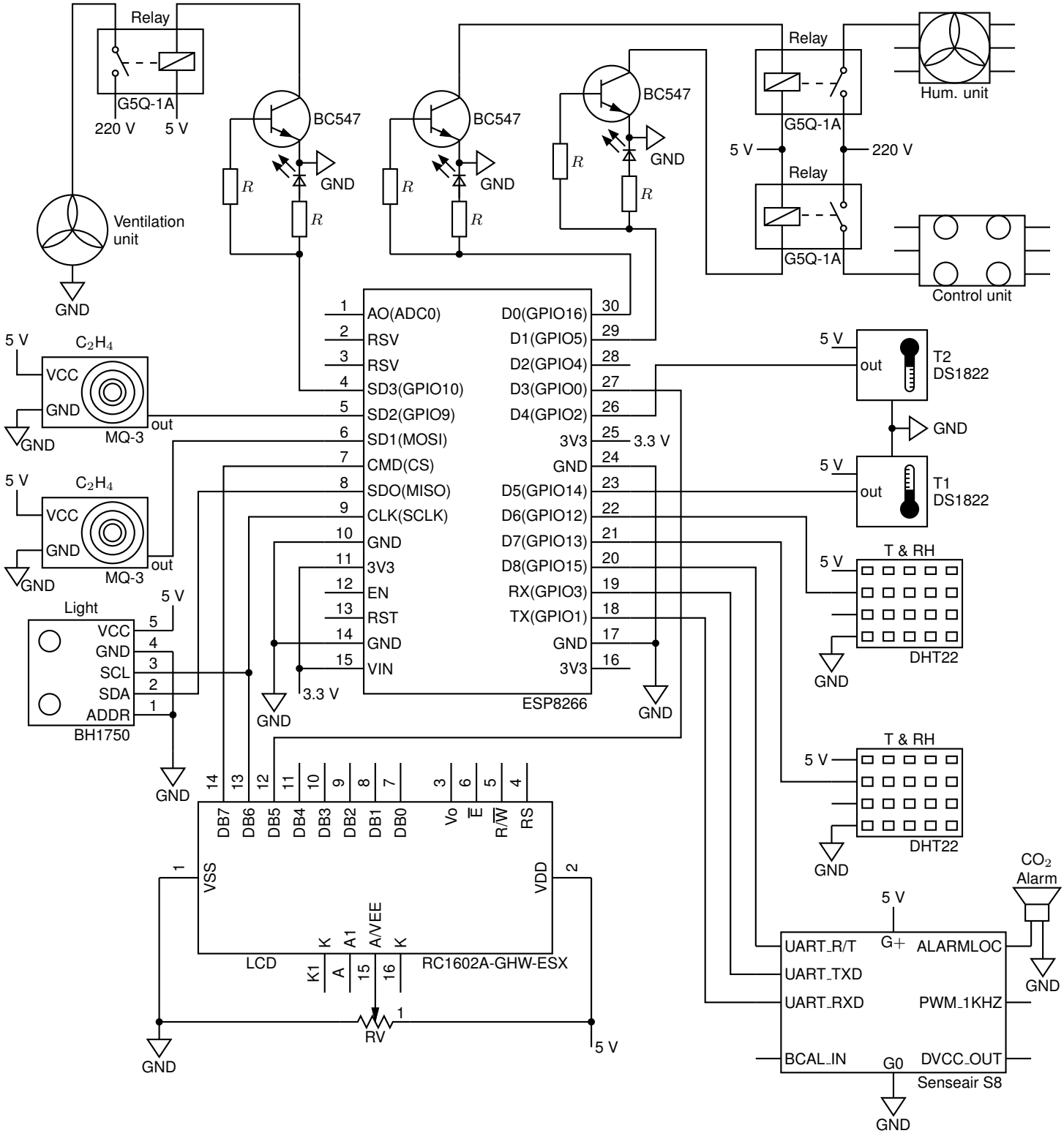


Figure 72: Part of a cold room controller, with some custom sensors [ColdControl1.m4].

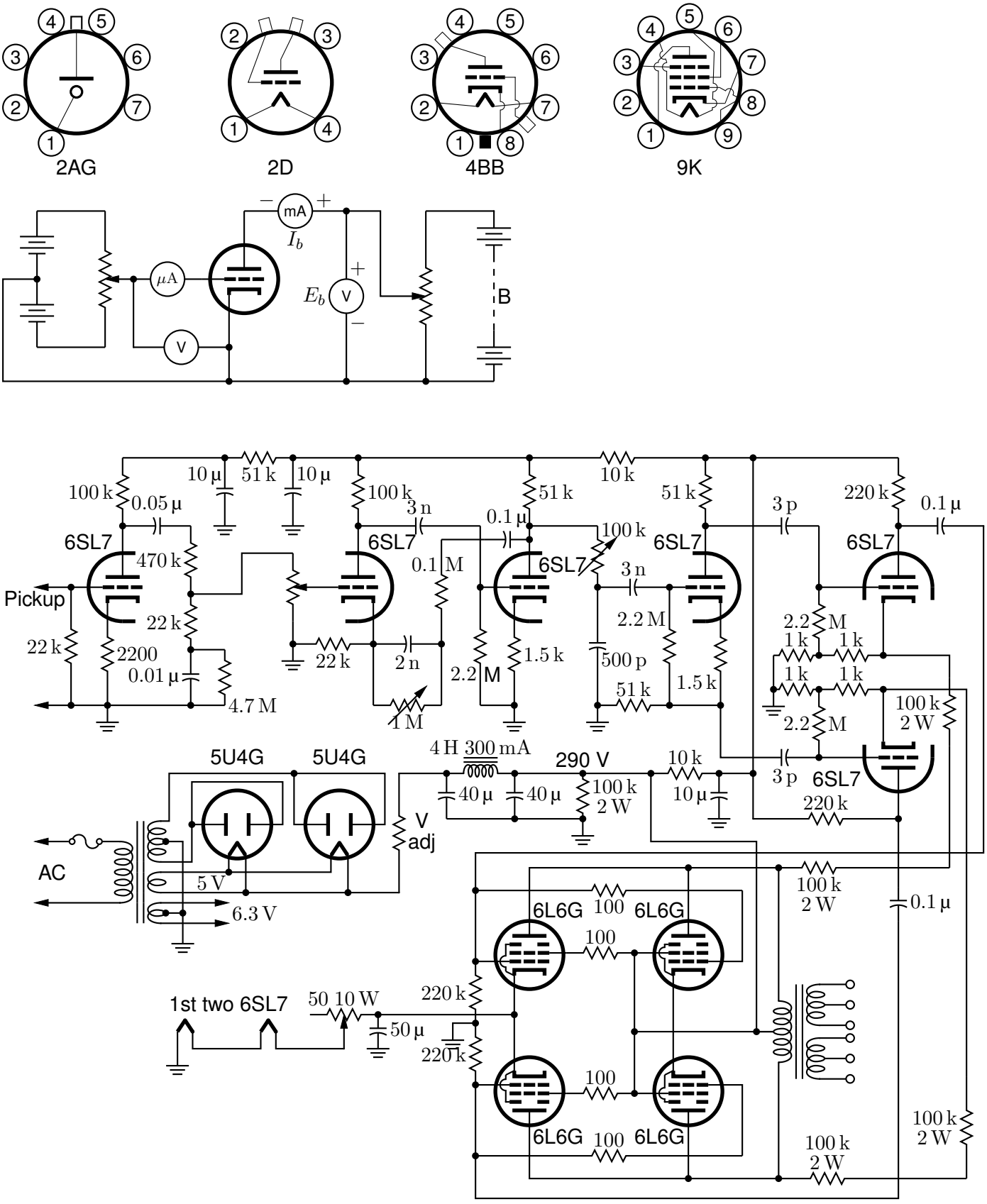


Figure 73: Electron-tube diagrams: a few bottom-view base diagrams, a generic triode test circuit, and a 25-watt audio amplifier adapted from F. Langford-Smith, *Radiotron Designer's Handbook,* fourth edition, Harrison, NJ: Radio Corporation of America, 1952 `[Tubediags.m4]`.

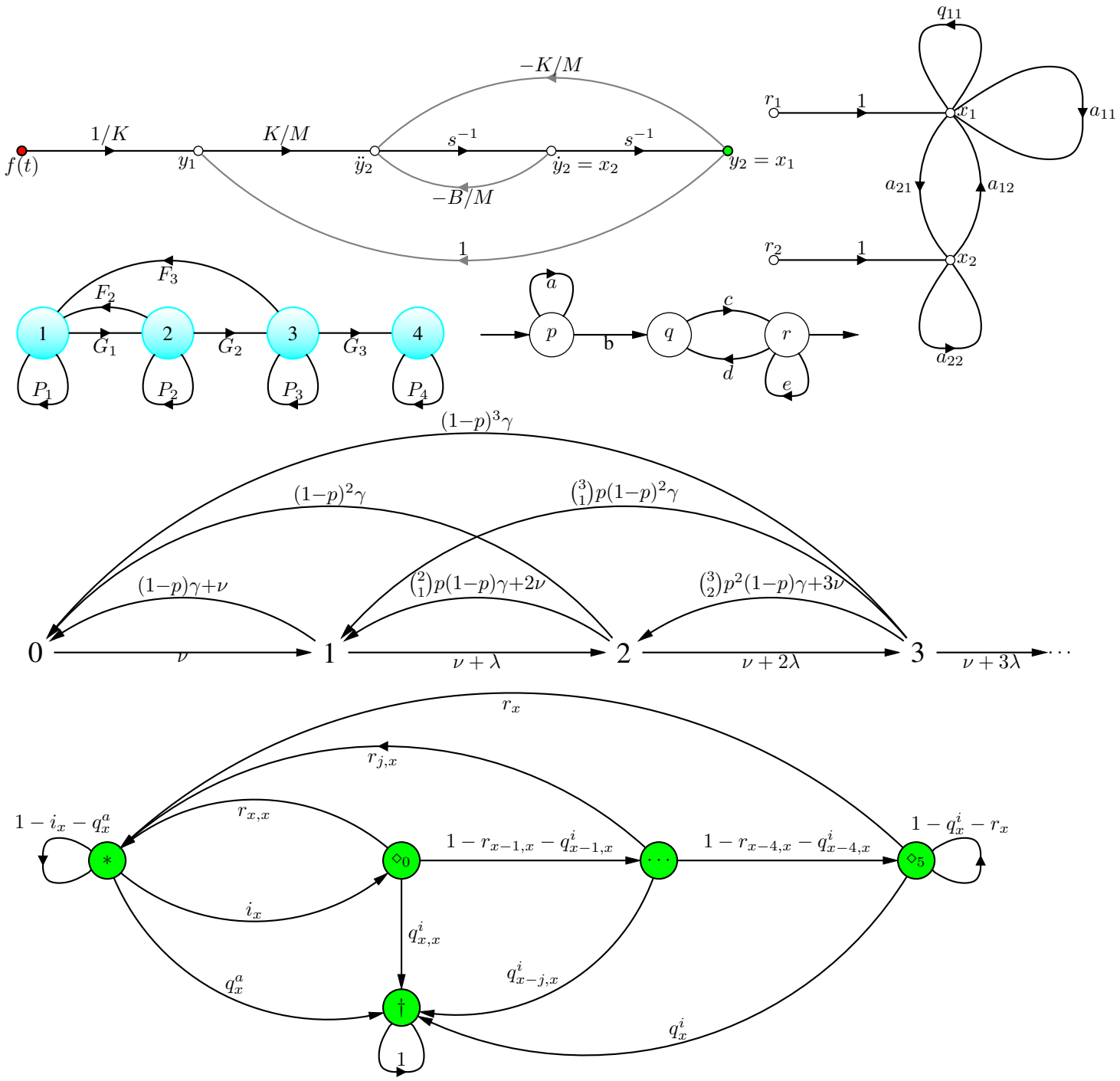


Figure 74: Signal-flow graphs `[sfg.m4]`.

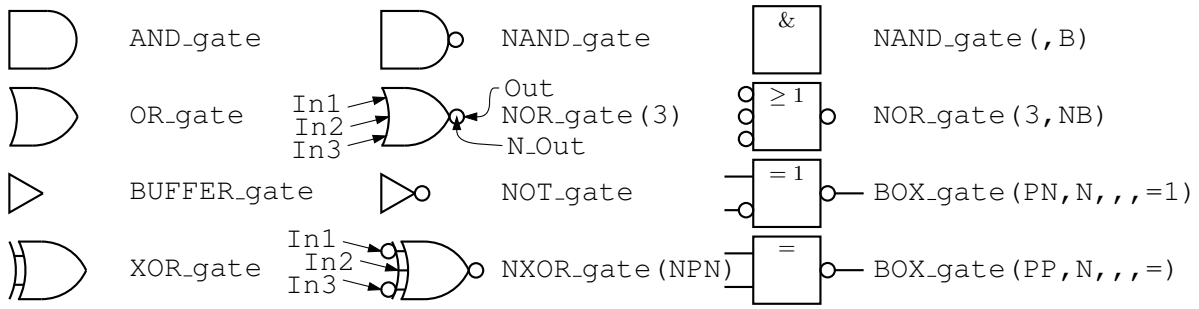


Figure 75: Basic logic gates [Logic.m4].

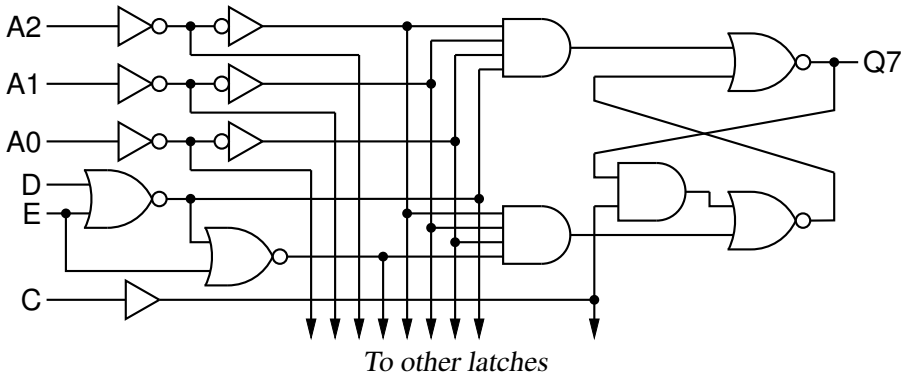


Figure 76: General-purpose latch: a small logic circuit [ex08.m4].

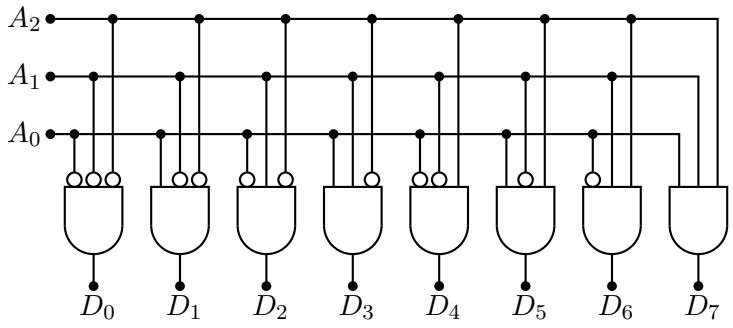


Figure 77: Decoder logic, constructed using the for_ macro [Decoder.m4].

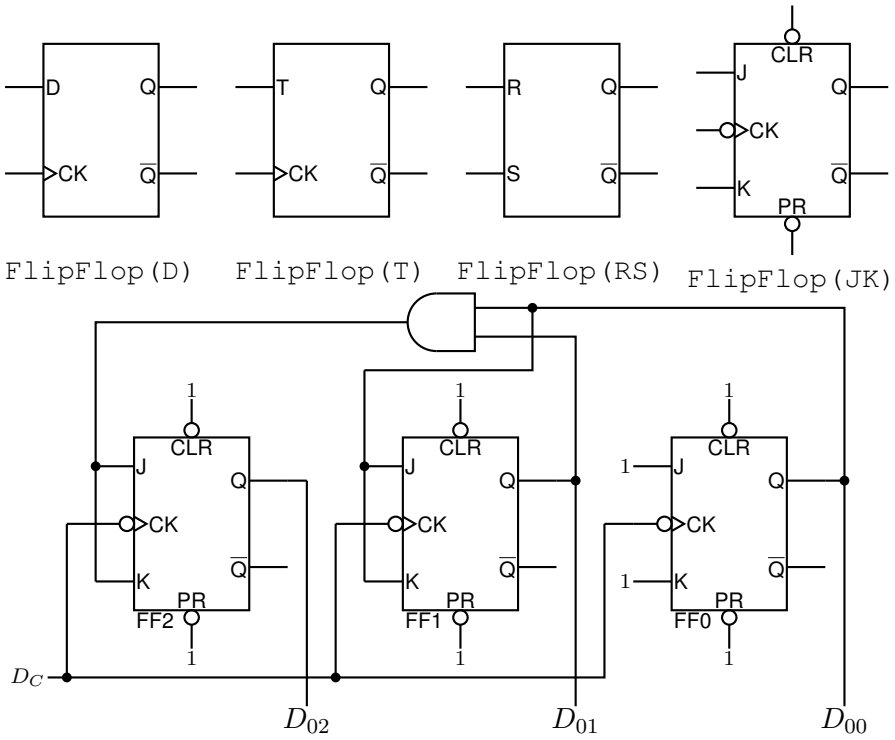


Figure 78: Some flip-flops [ex21.m4].

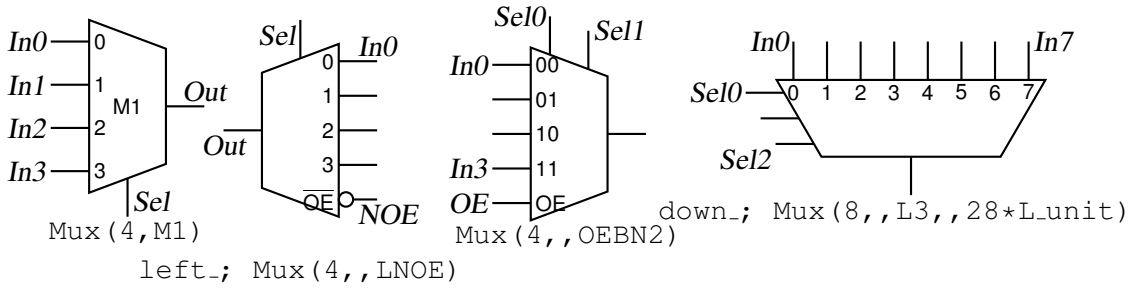


Figure 79: Multiplexer [Multiplexer.m4].

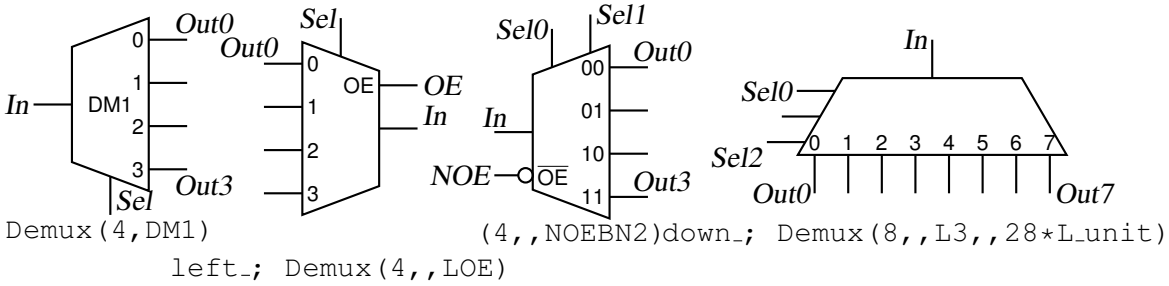


Figure 80: Demultiplexer [Demultiplexer.m4].

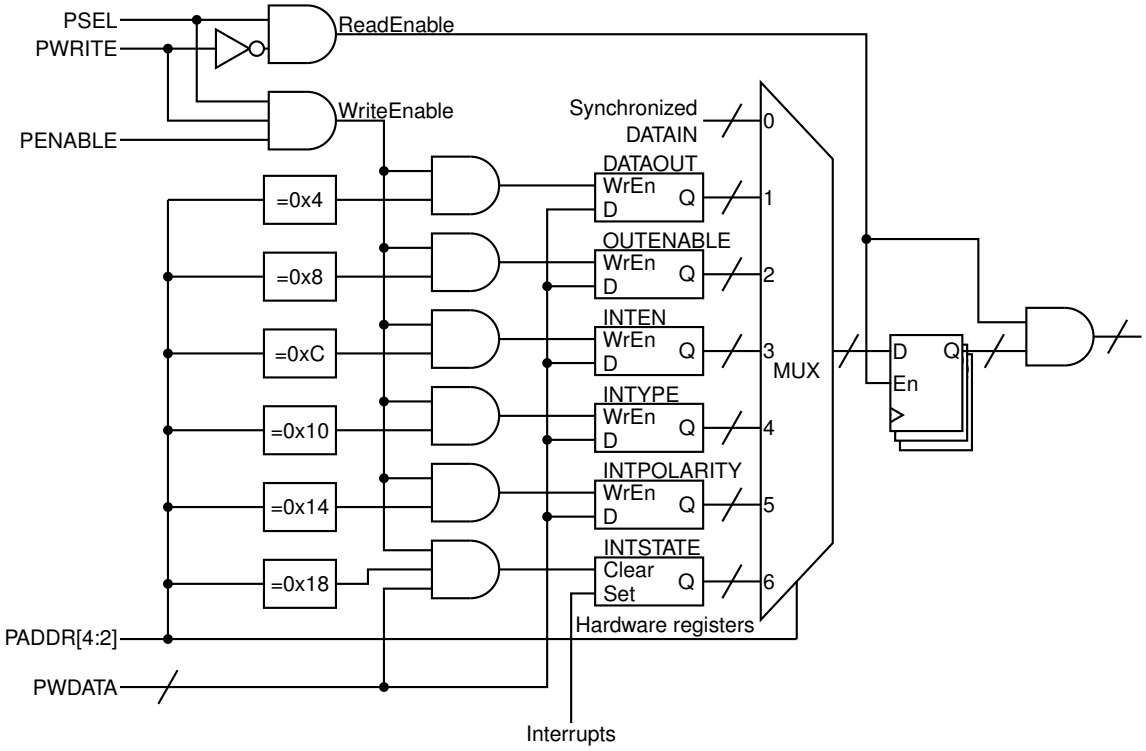


Figure 81: Multiplexer circuit `[Muxcct.m4]`.

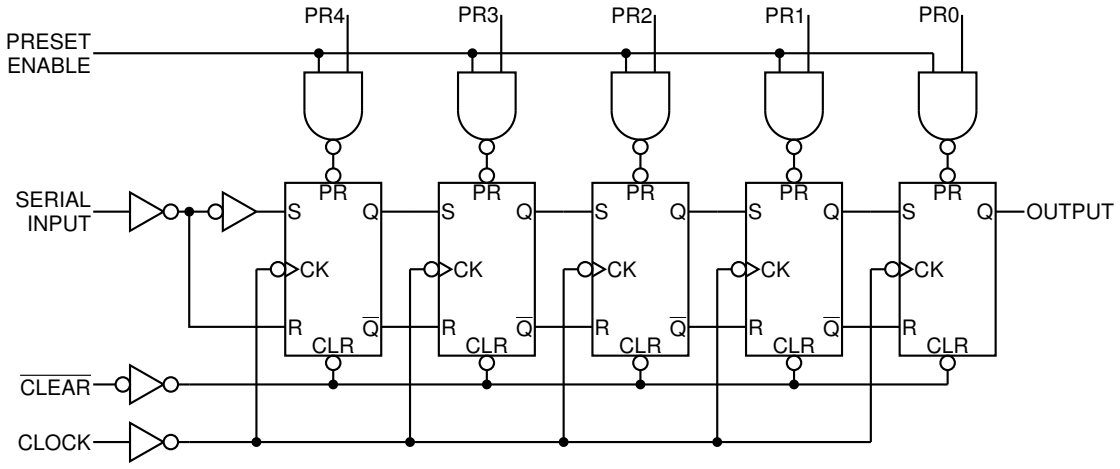


Figure 82: A 5-bit shift register drawn using a custom flip-flop `[ShiftR.m4]`.

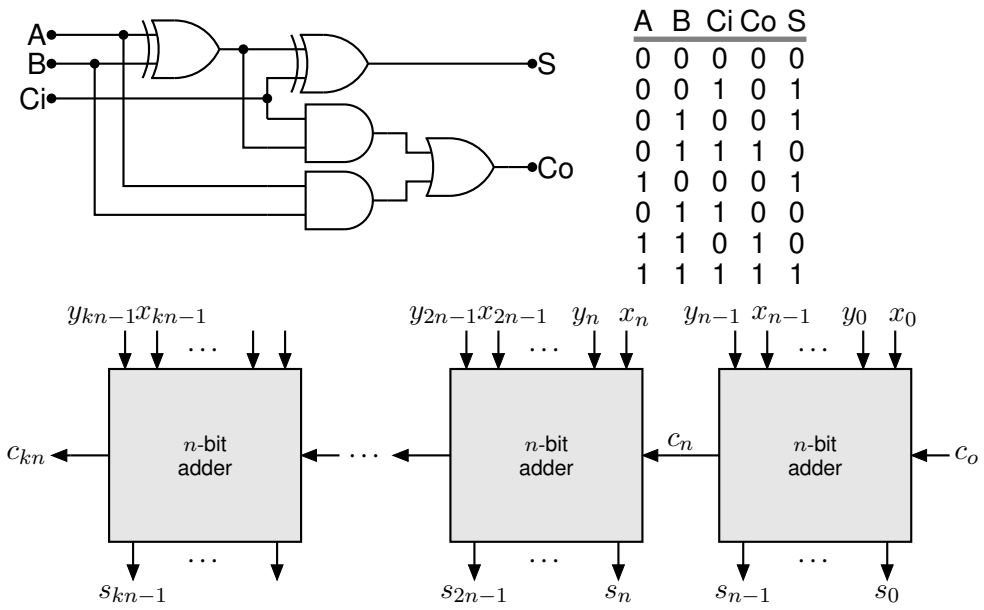


Figure 83: A full adder and a cascade of n-bit adders `[Adder.m4]`.

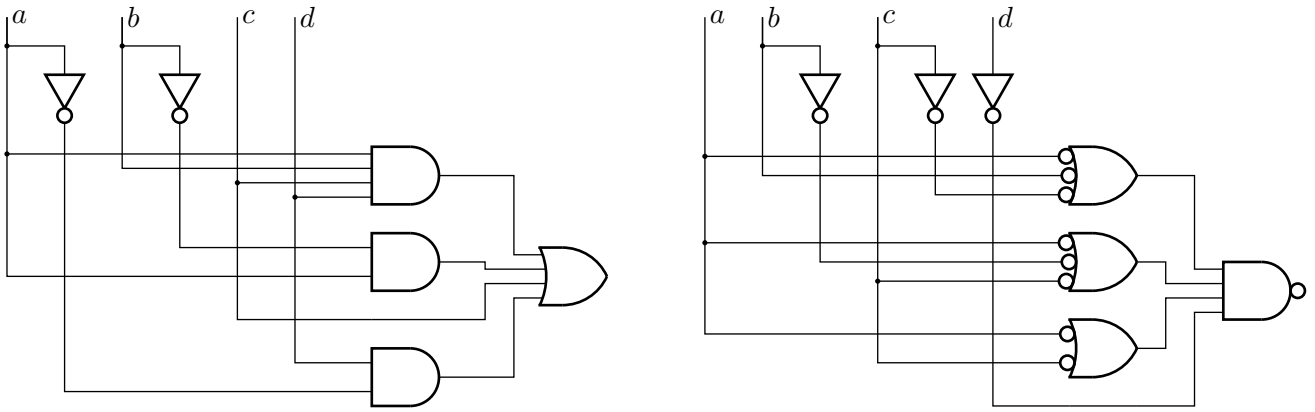


Figure 84: A way of automatically drawing two-layer logic diagrams `[CanLogic.m4]`.

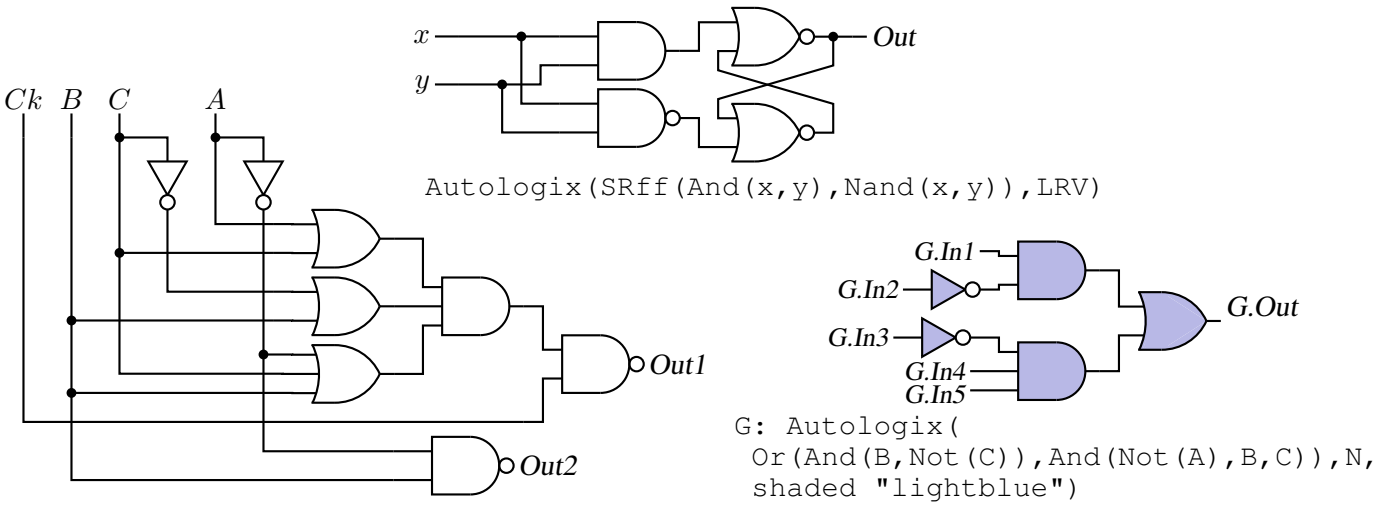


Figure 85: The `Autologix(`*Boolean expression*; *Boolean expression...* , *options*`)` macro automatically draws Boolean expressions in function notation. The function tree is drawn, then a row or column of inputs, then the connections. The default result is on the left, a custom element at the top, and a tree of gates only is shown on the right. `[Alogix.m4]`.

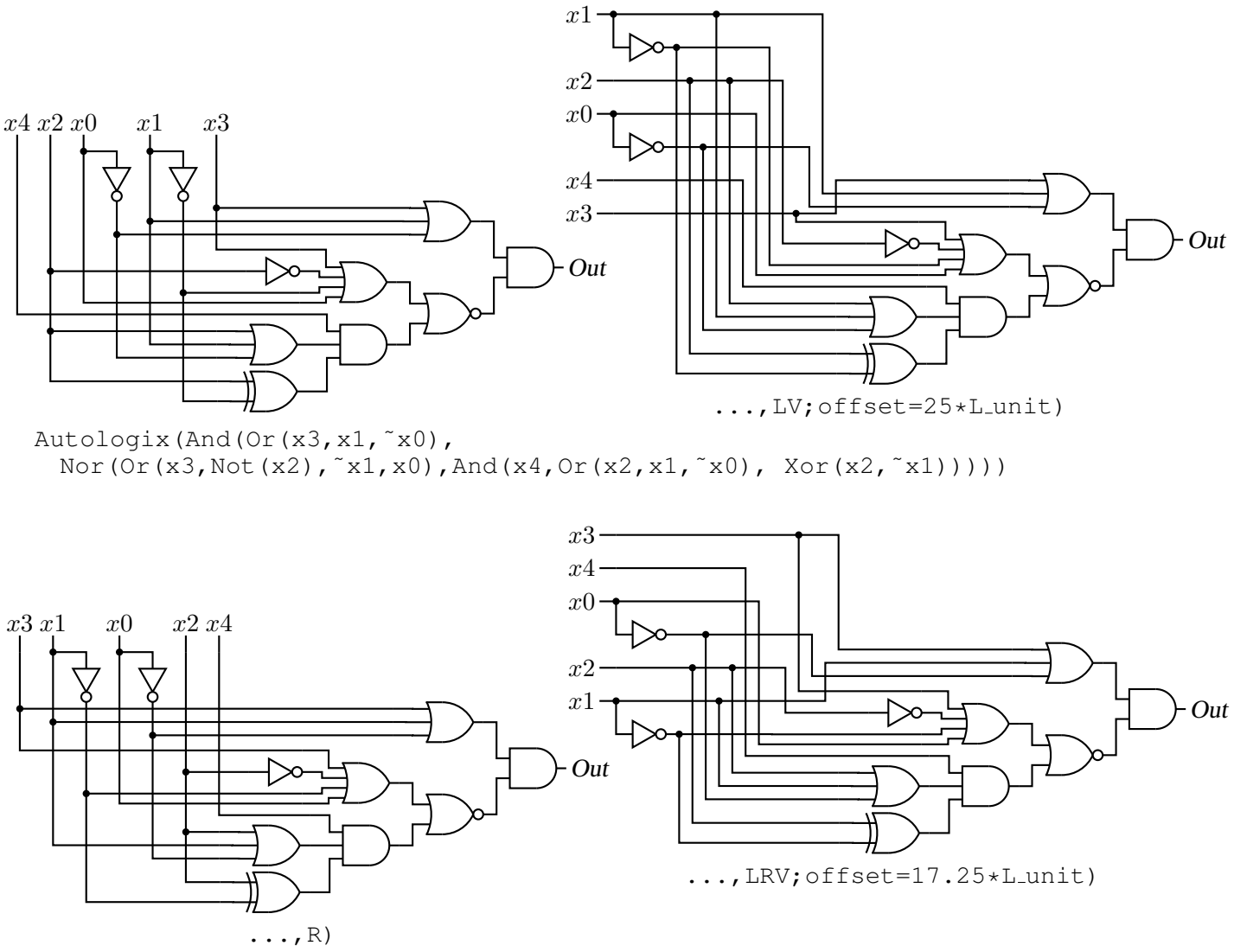


Figure 86: The `Autologix` macro can draw inputs on the left but the added drawing complexity may require hand tuning with second-argument options: `L` puts the inputs on the left, `R` reverses their order, `V` scans the input arguments in reverse order, and `offset=`*value* displaces the array of inputs `[ABlogix.m4]`.

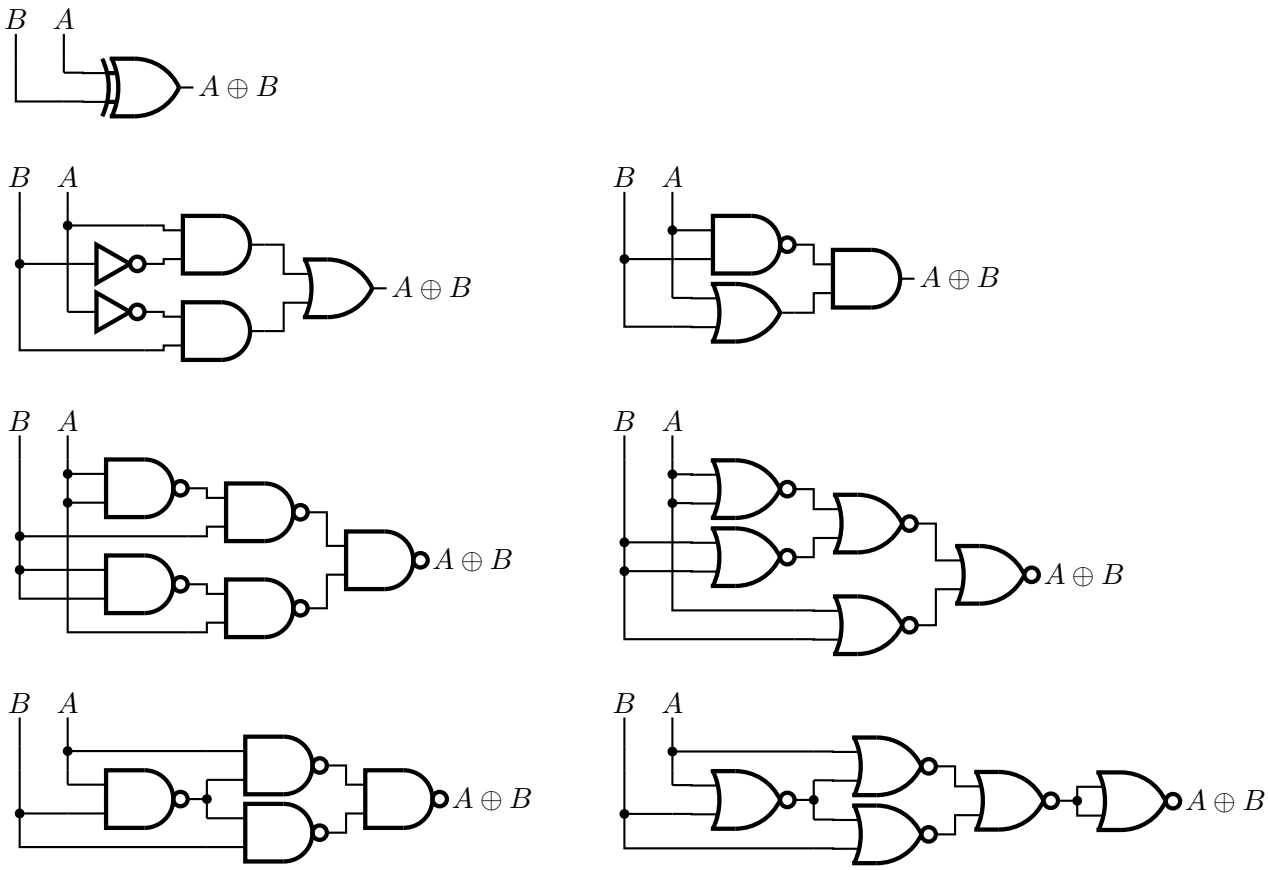


Figure 87: Realizations of the XOR function using `Autologix [XOR.m4]`.

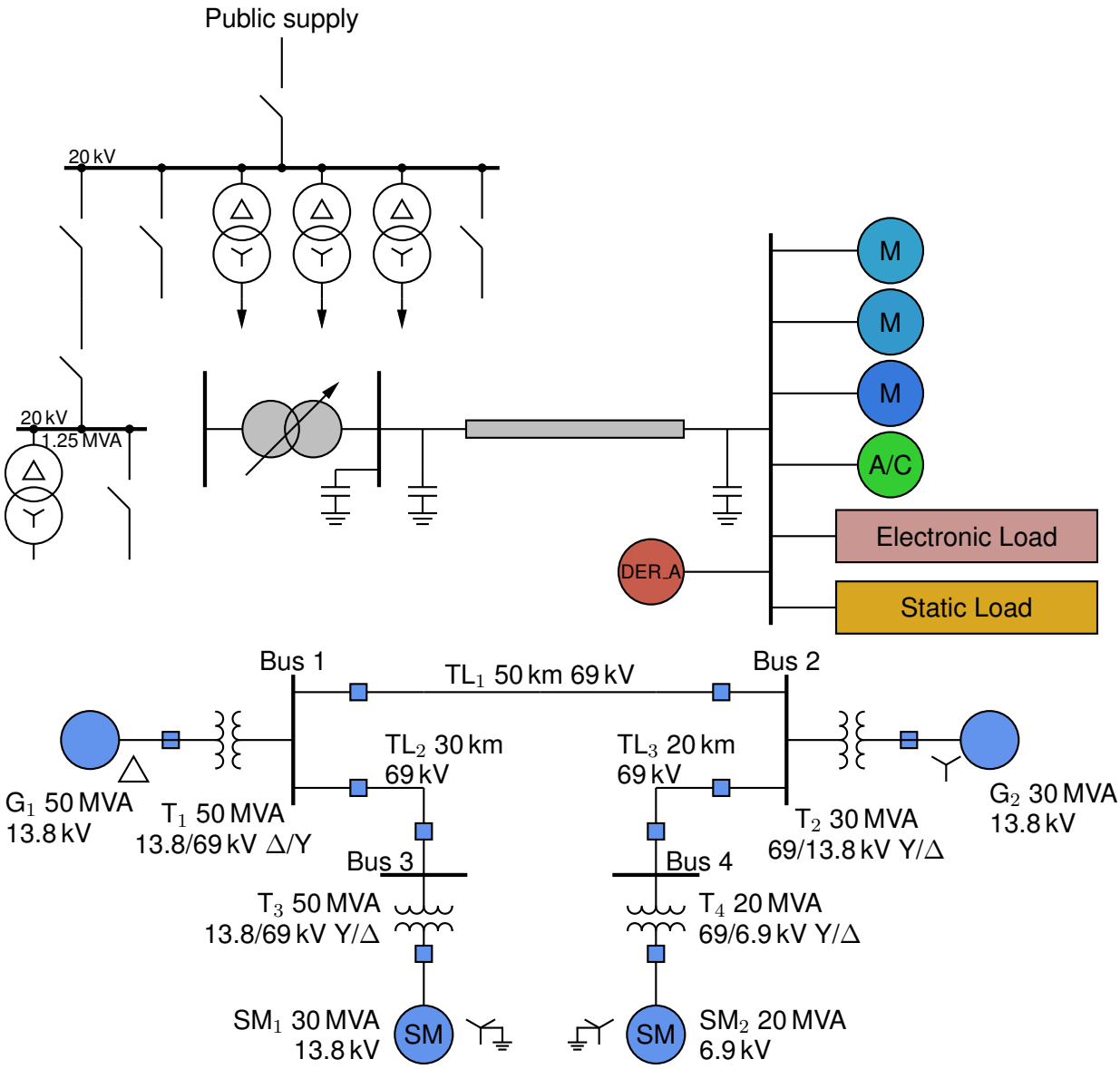


Figure 88: A test of single-line diagram macros `[EEP.m4]`.

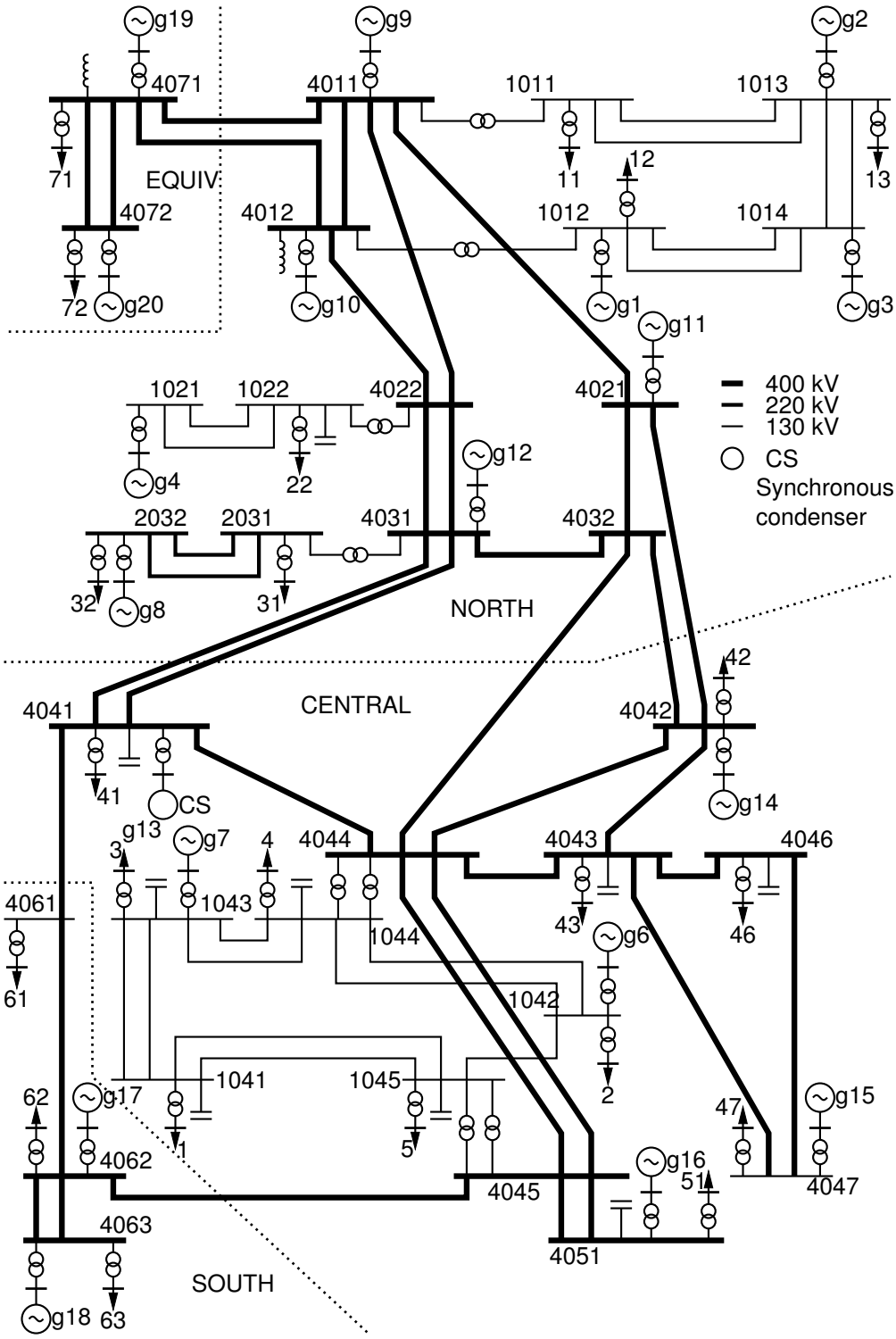


Figure 89: The Nordic test system `[Nordic.m4]`.

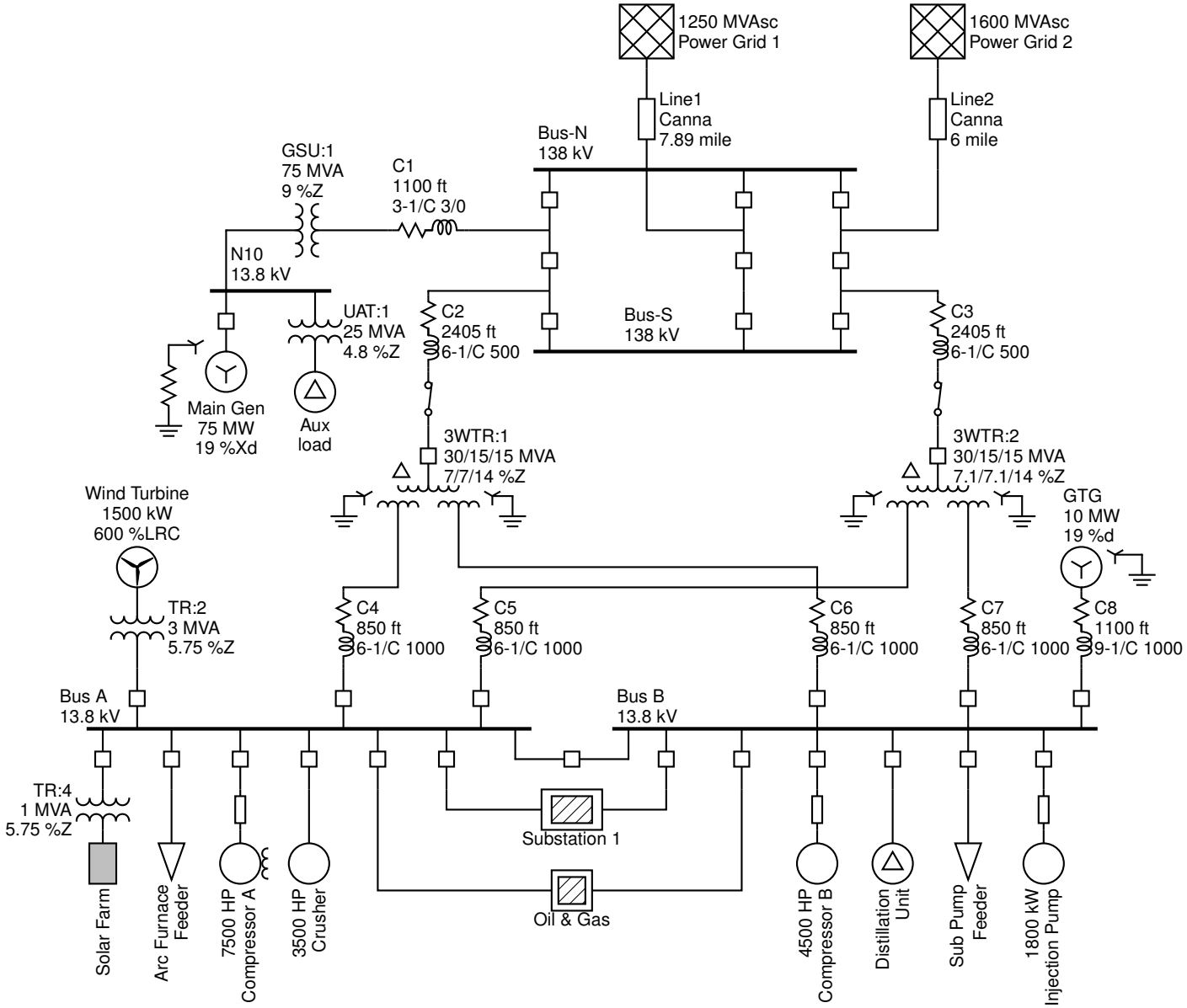


Figure 90: A single-line example for performing load-flow analysis `[Loadflow.m4]`.

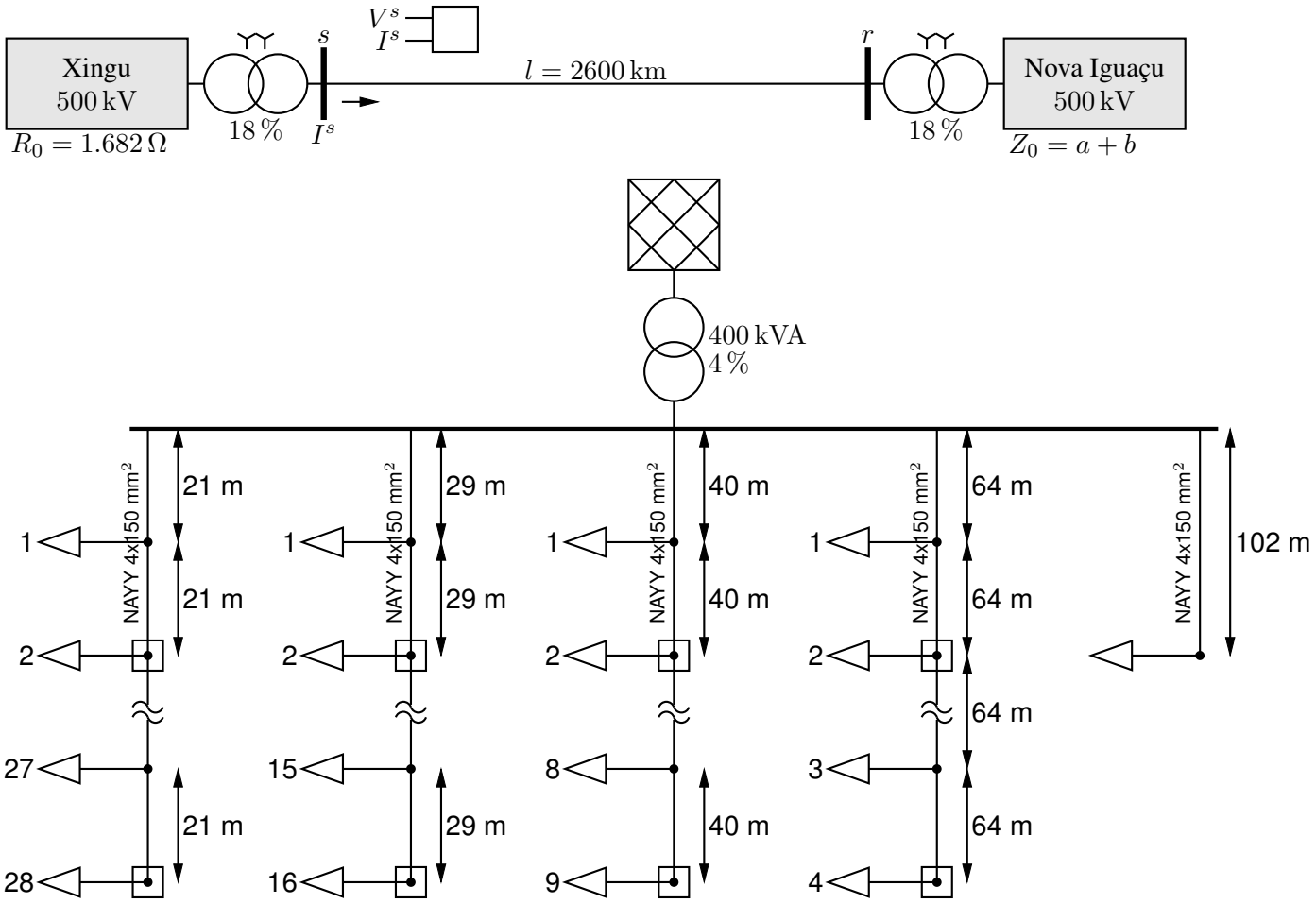


Figure 91: Another single-line distribution diagram [OneLine.m4].

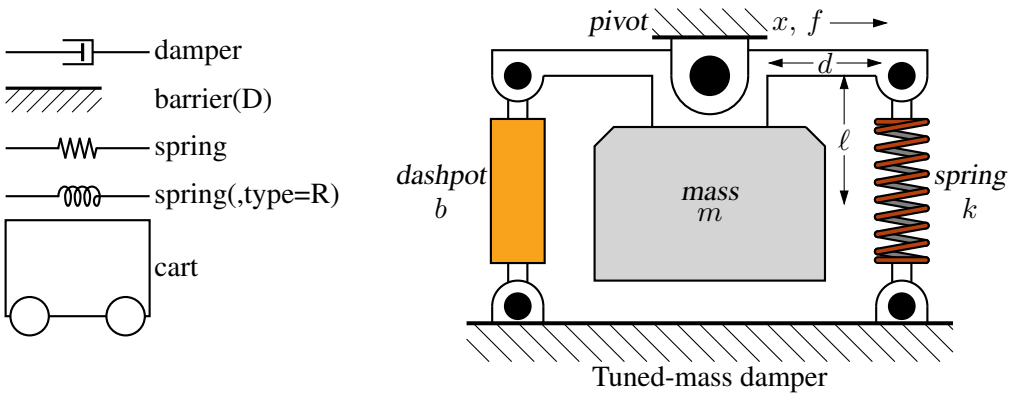


Figure 92: Some basic mechanical components and an example containing a custom spring [Mechanics.m4].

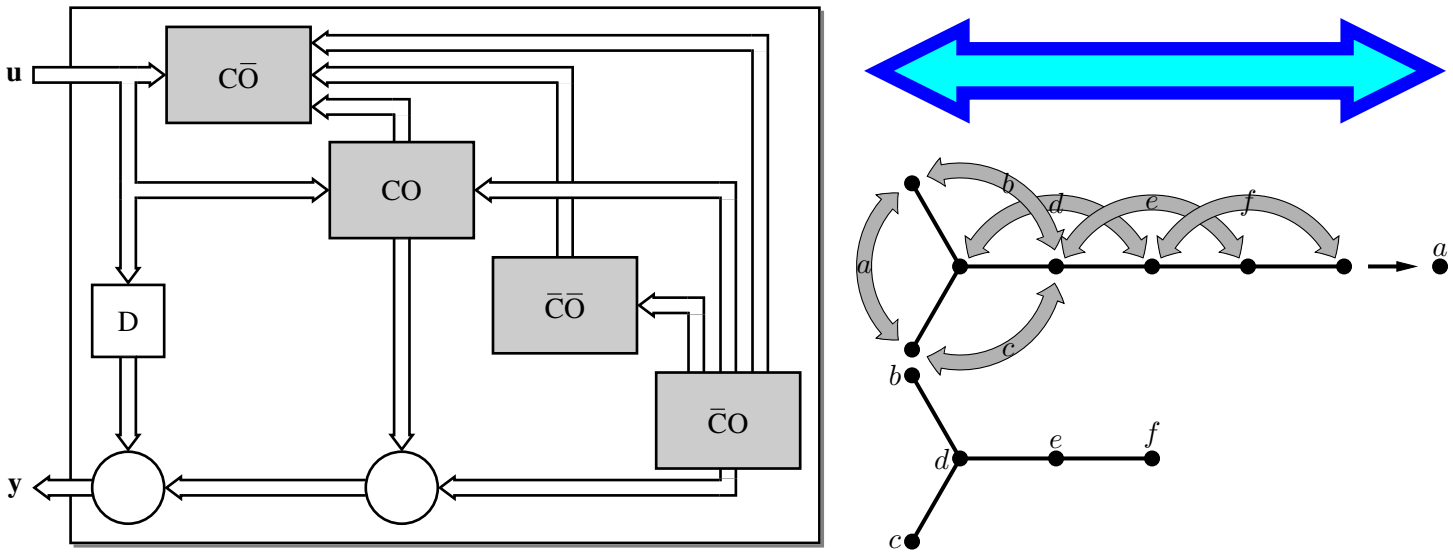


Figure 93: Use of `darrow` and `Darc` `[ex05.m4]`.

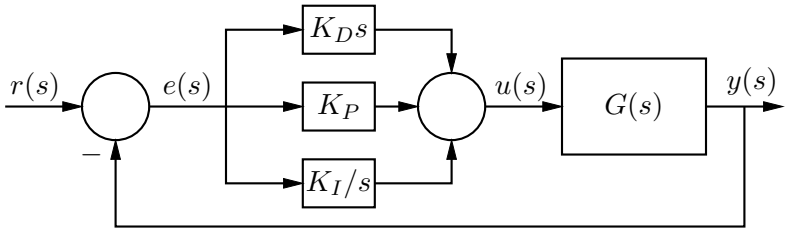


Figure 94: PID control `[conPID.m4]`.

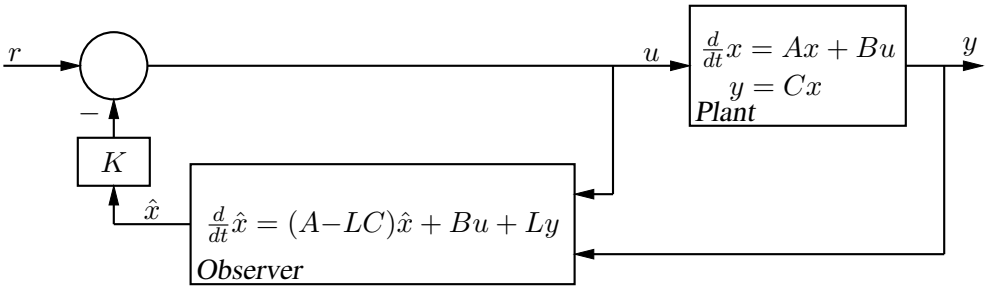


Figure 95: Full-order observer feedback `[conFullOrd.m4]`.

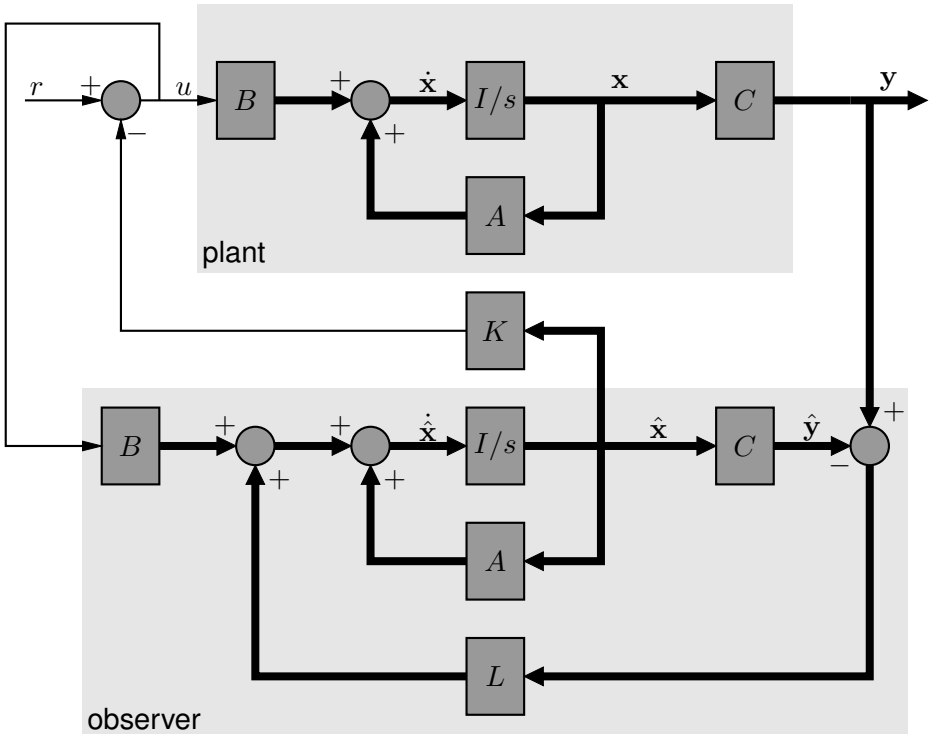


Figure 96: Single-input plant with feedback from a full-order observer `[conSIMO.m4]`.

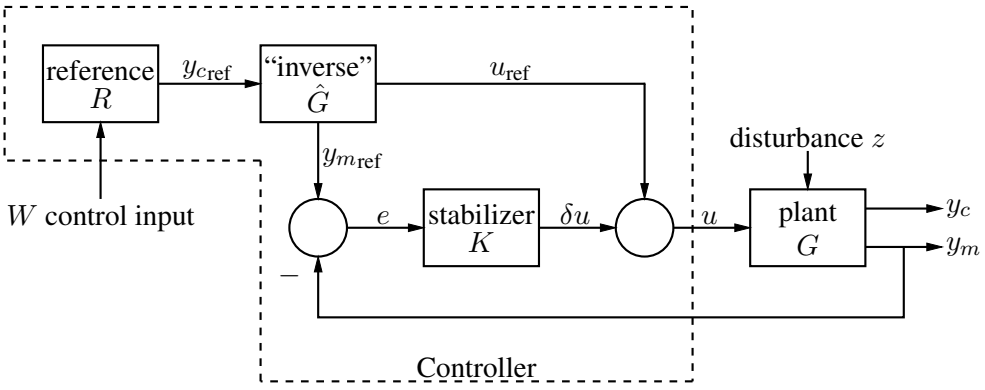


Figure 97: Nonlinear feedforward (for performance) and small-signal feedback (for stability) `[conNL.m4]`.

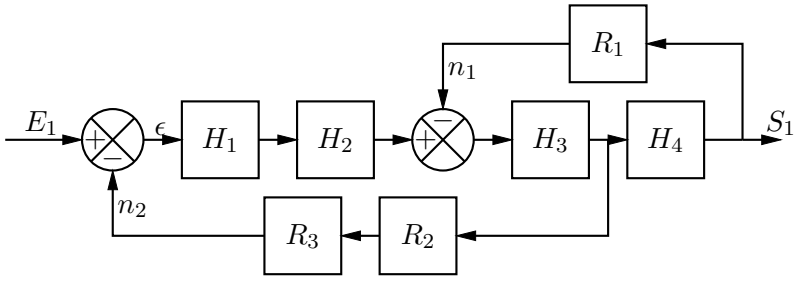


Figure 98: A multi-block example `[conMB.m4]`.

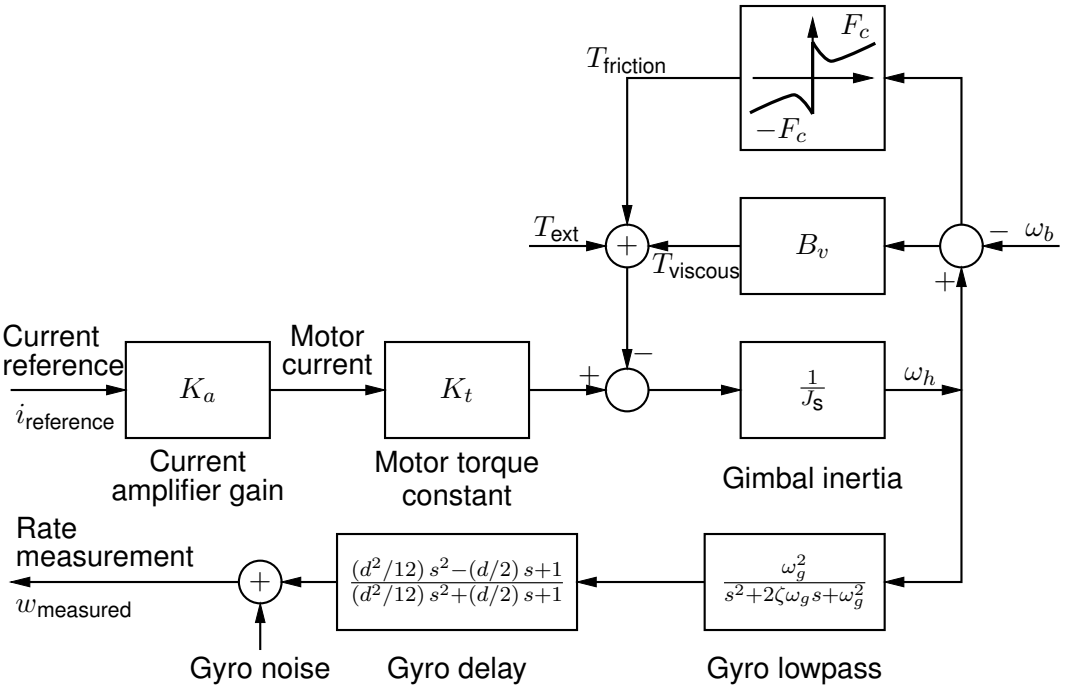


Figure 99: Block diagram of a single-axis gimbal `[conGimbal.m4]`.

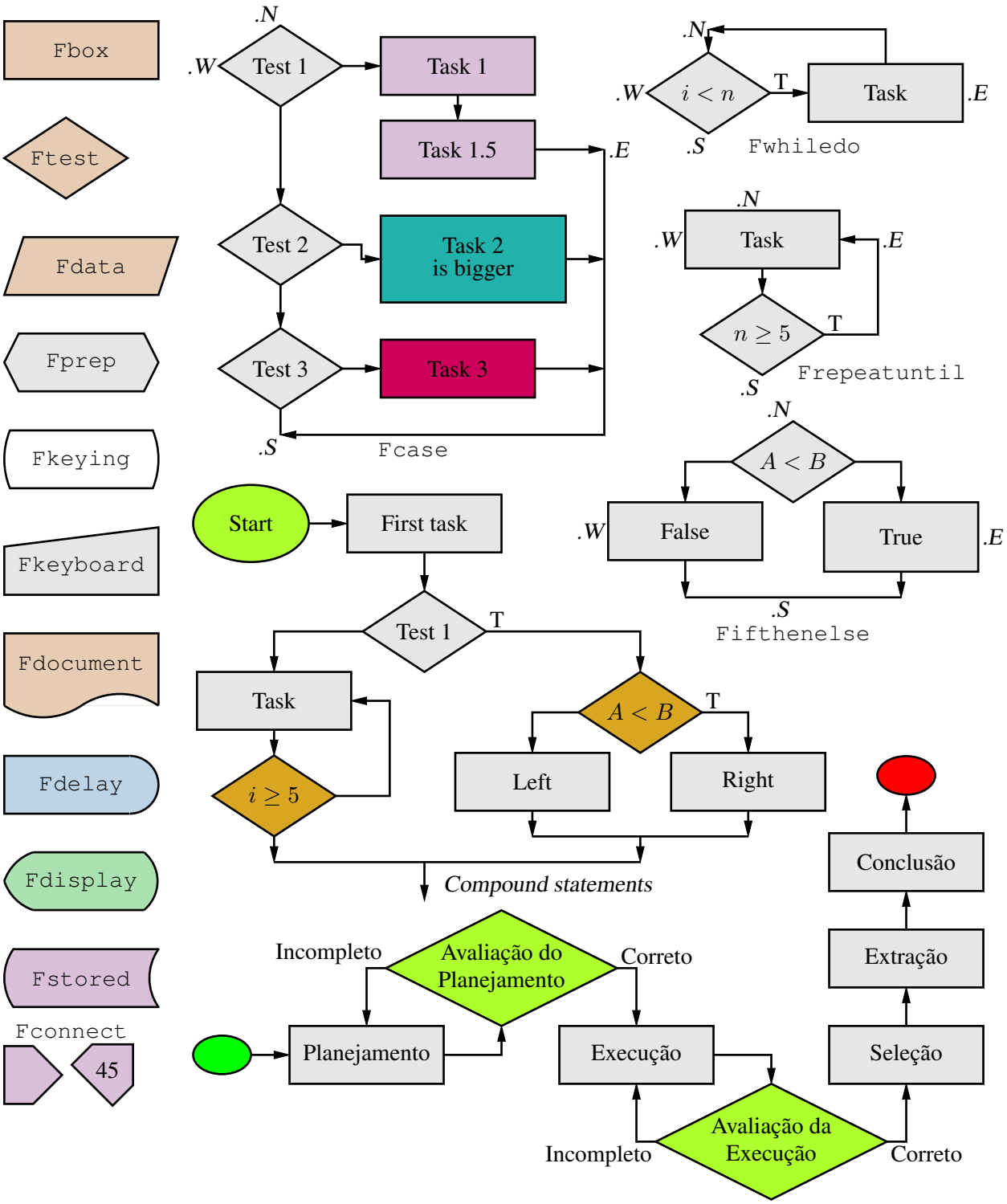


Figure 100: A flowchart sampler `[Flow.m4]`.

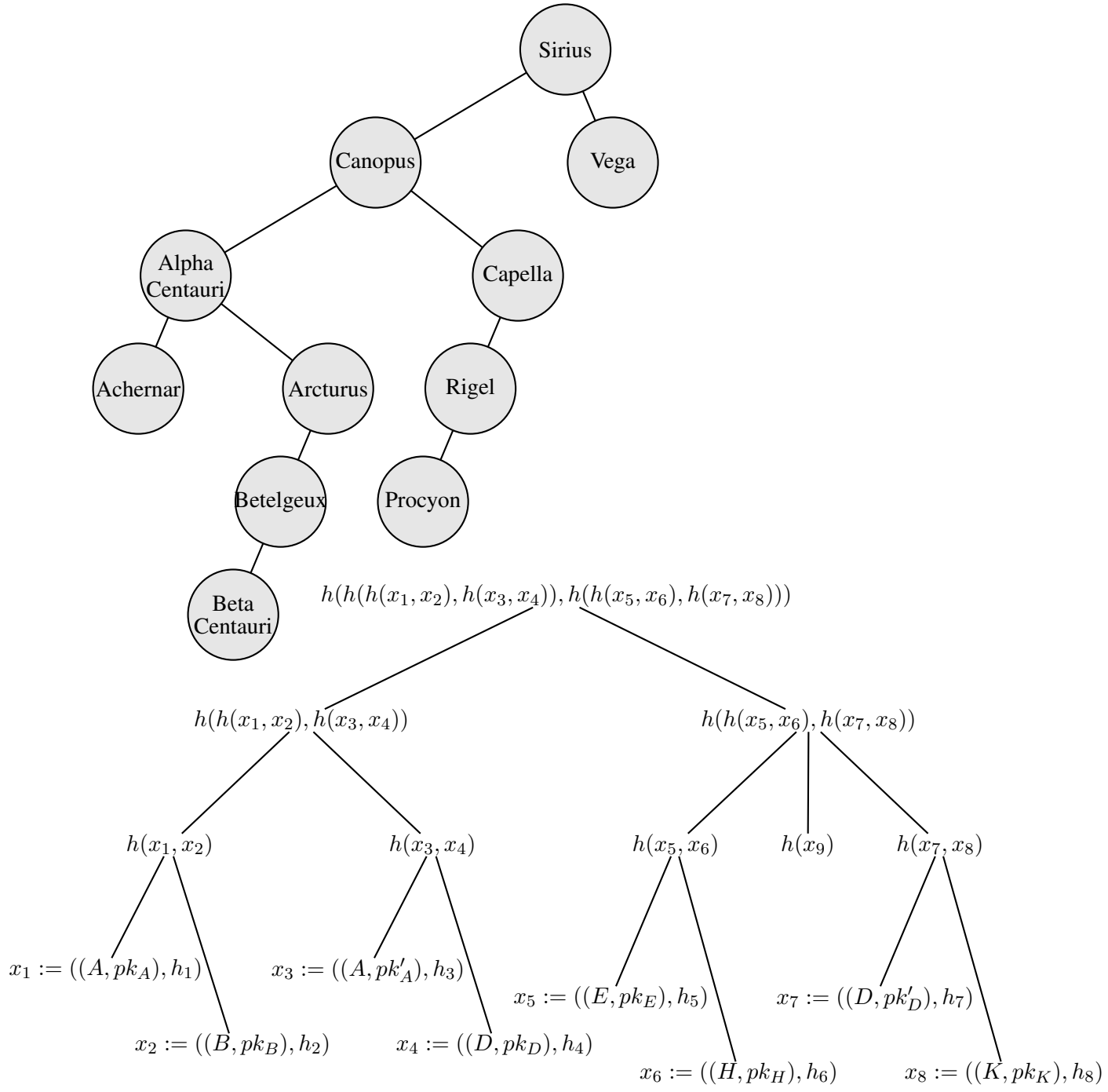


Figure 101: Trees `[Btree.m4]`.

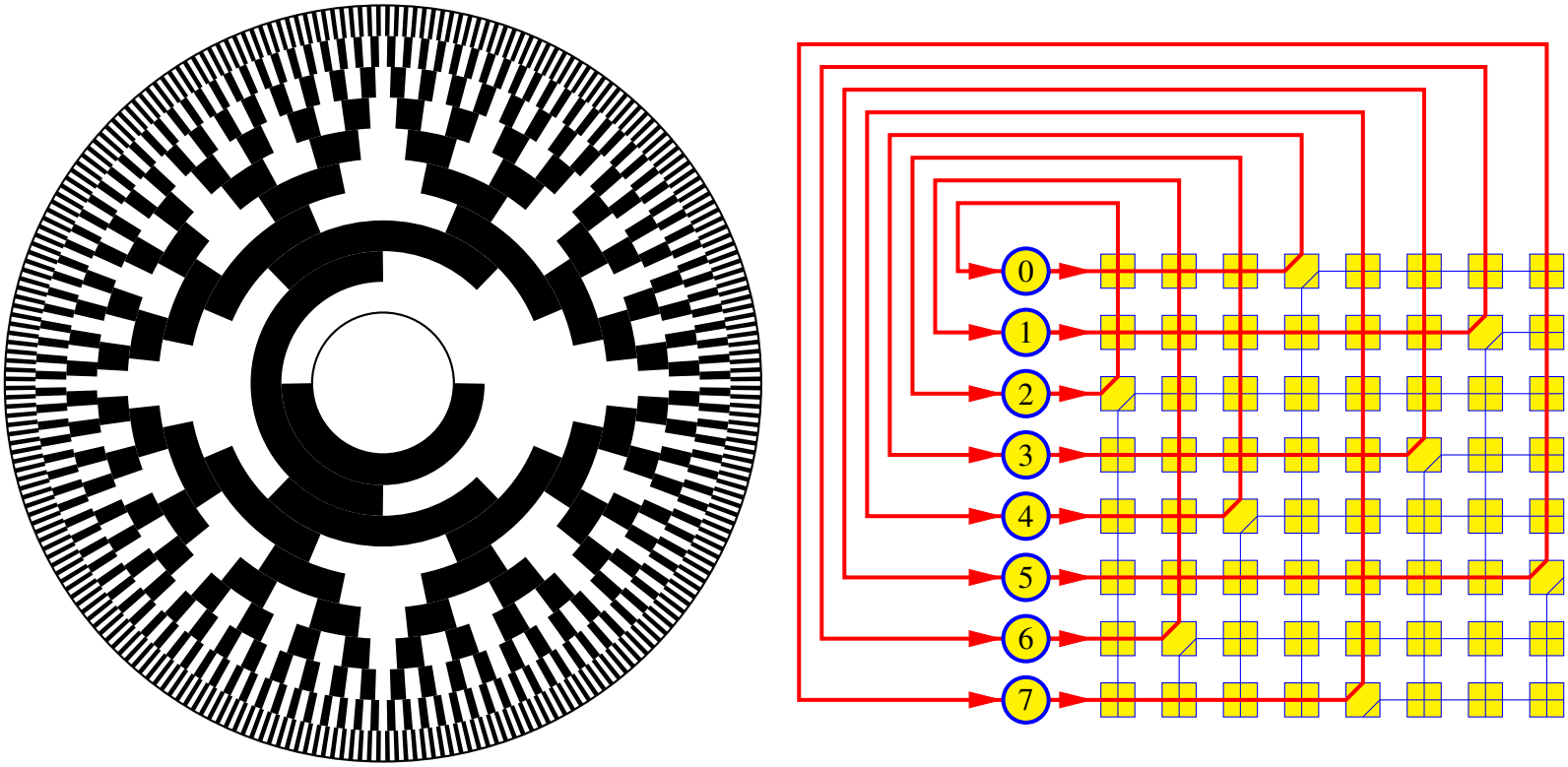


Figure 102: The power of looping and branching: Gray code 10-bit encoder disk pattern and a crossbar switch `[GrayCode.m4]`.

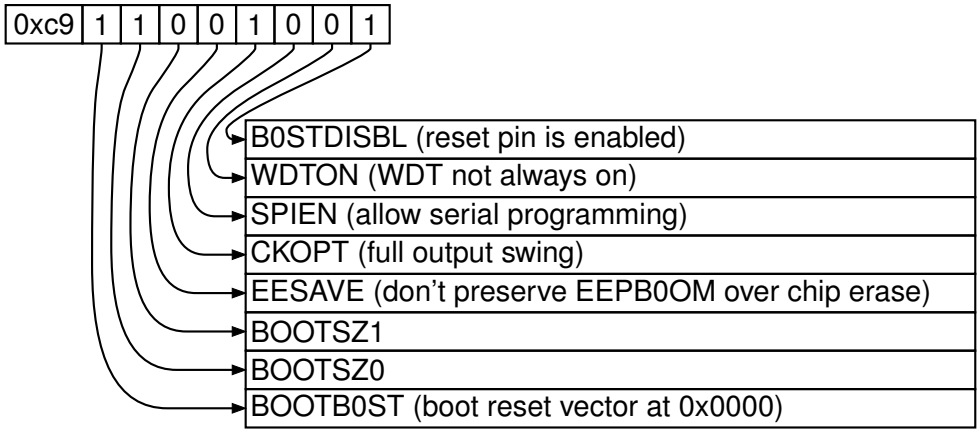


Figure 103: Elementary splines `[Byte.m4]`.



Figure 104: A customizable seven-segment display showing the numbered segments, a custom shape, and the numerals from 0 to 9 `[Sevensegment.m4]`.

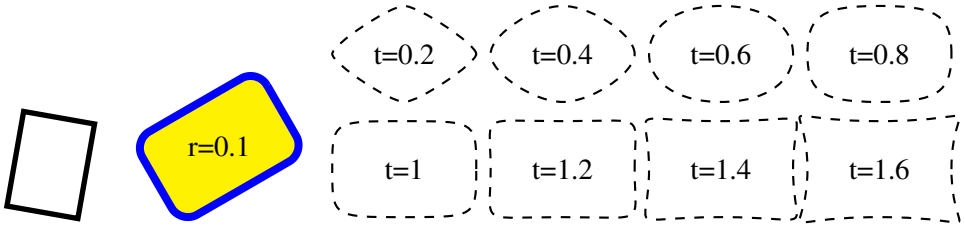


Figure 105: The macro `rotbox(`*wid,ht,type*`,[r|t=`*val*`])` draws a box in the current direction `[Rotbox.m4]`.

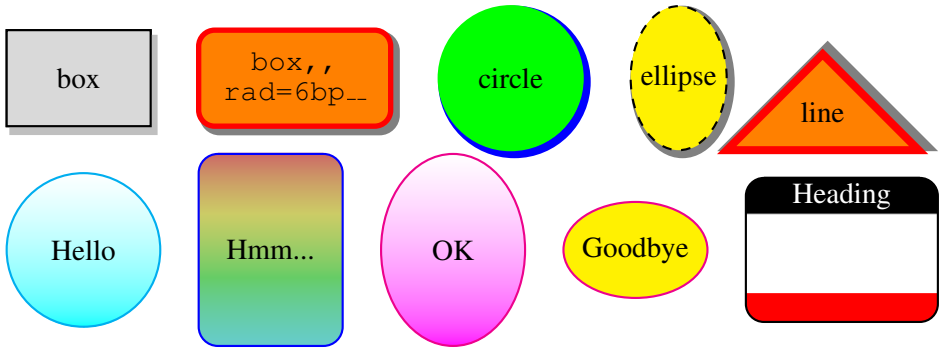


Figure 106: Embellishments: the top row illustrates `shadowed(box|circle|ellipse|line,[at` *position*`],`*keys*`)` and the second row `ColoredV(box|circle|ellipse,(`*r,g,b*`)|((`*colorseq*`)),`*attributes*`)` `[Shadowed.m4]`.

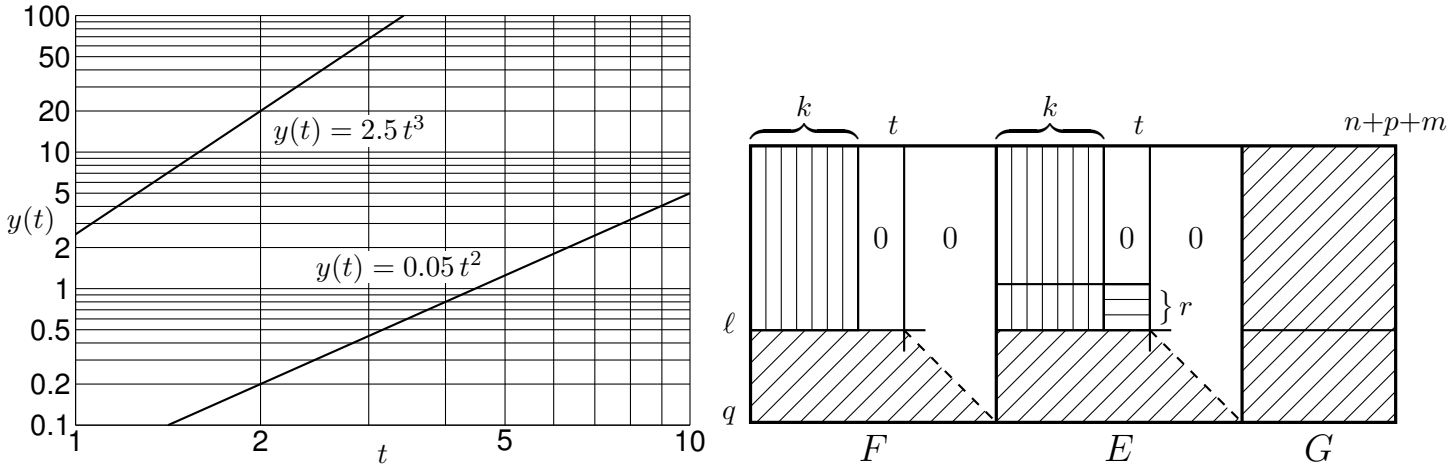


Figure 107: A graph and crosshatching example drawn using the pic language `[Loglog.m4]`.

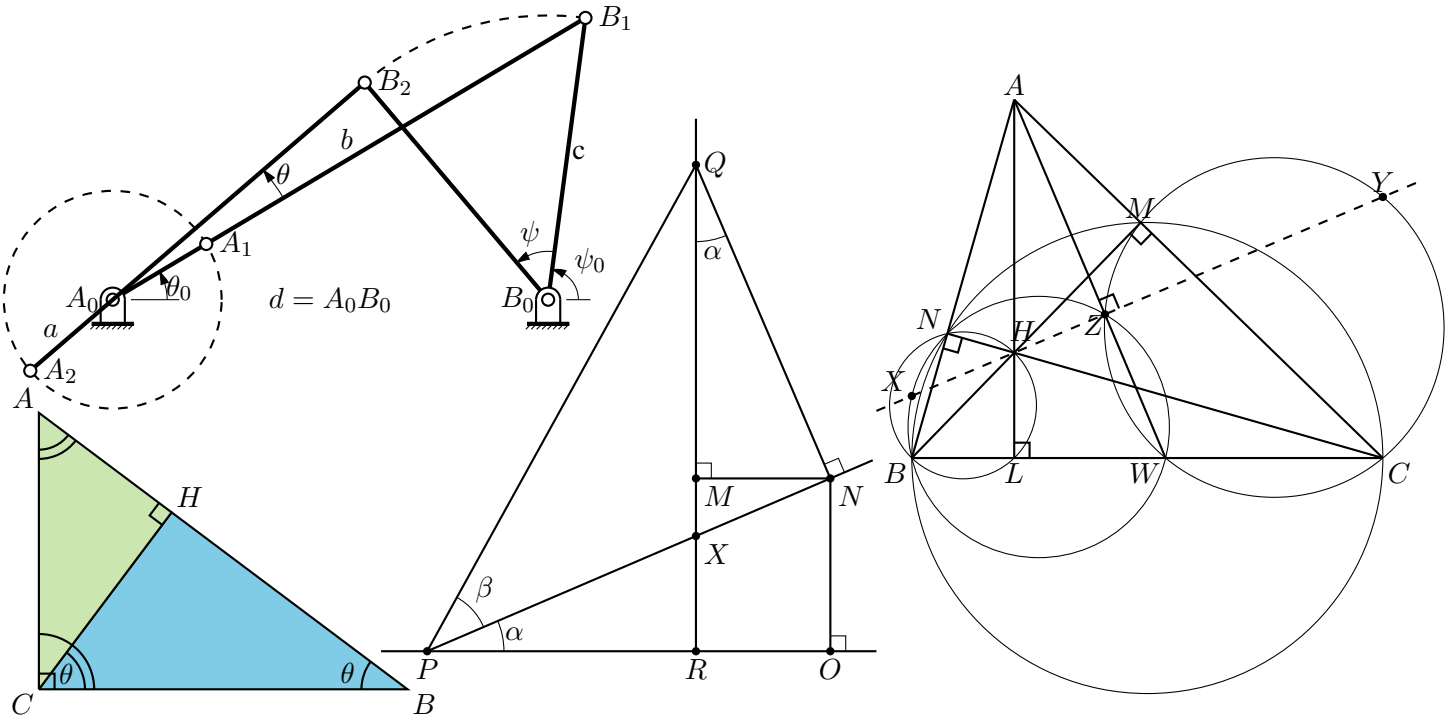


Figure 108: Some geometrical constructions `[Geometry.m4]`.

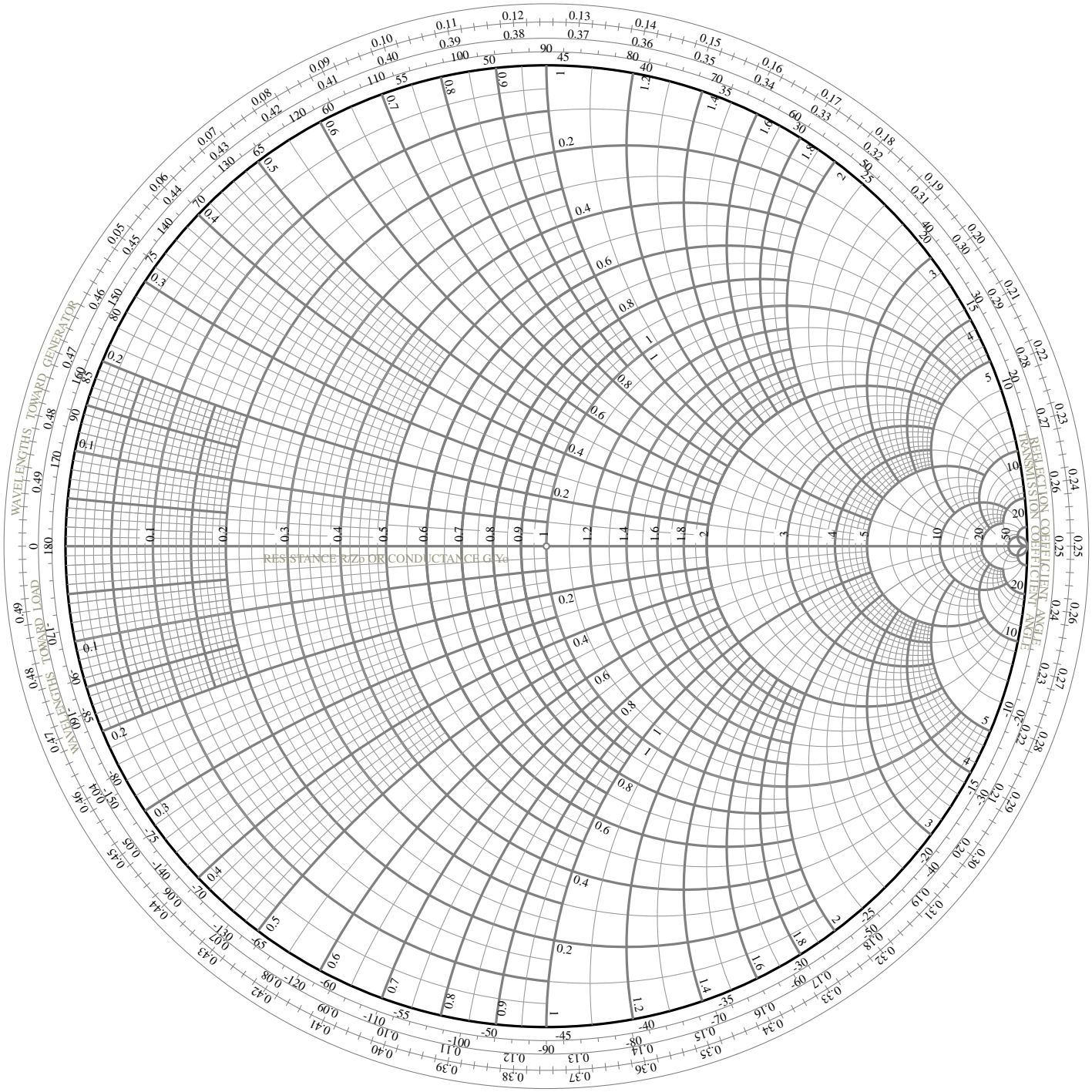
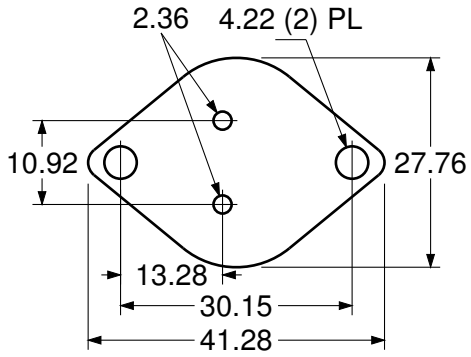
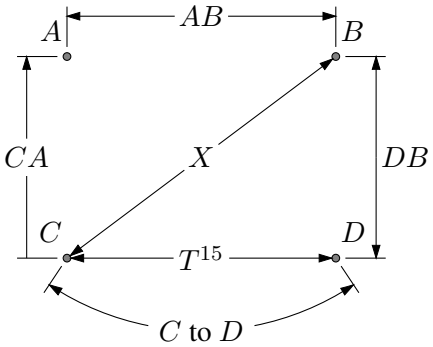
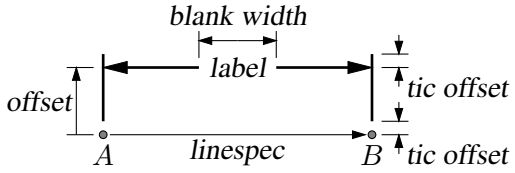


Figure 109: A Smith chart [Smithlchart.m4].








```
dimension_(from A to B,0.3,$AB$,20bp__)
dimension_(from C to B,,$X$,16bp__)
dimension_(from C to A,0.3,$CA$,14bp__,,->)
dimension_(from D to B,-0.3,"$DB$" ljust)
dimension_(from C to D,,s_box($T^{%g}$,15),W)
arcdimension_(from C to D with .c at \
  0.5 between A and B,12bp__,s_box($C$ to $D$),W)
dimension_(from A to B,0.5,\sl label,29bp__,0.1)
```

Figure 110: Illustrating the macro `dimension_(`*linespec*, *offset*, *label*, `D|H|W|`*blank width*, *tic offset*,`<-|->)`. A negative second argument implies an offset to the right of the *linespec* direction. A *label* starting with `"` or `sprintf` is copied literally. If *label* is an `s_box(...)` then setting argument 4 to `H`, `W`, or `D` tailors the blank width to the `s_box` height, width, or diagonal respectively; i.e., `W` is equivalent to `s_wd+textoffset*2`. The macro `arcdimension_` is similar but the first argument specifies the arc to be dimensioned and the second argument is the outward radial offset of the dimension arrow arc. `[ex09.m4]`.

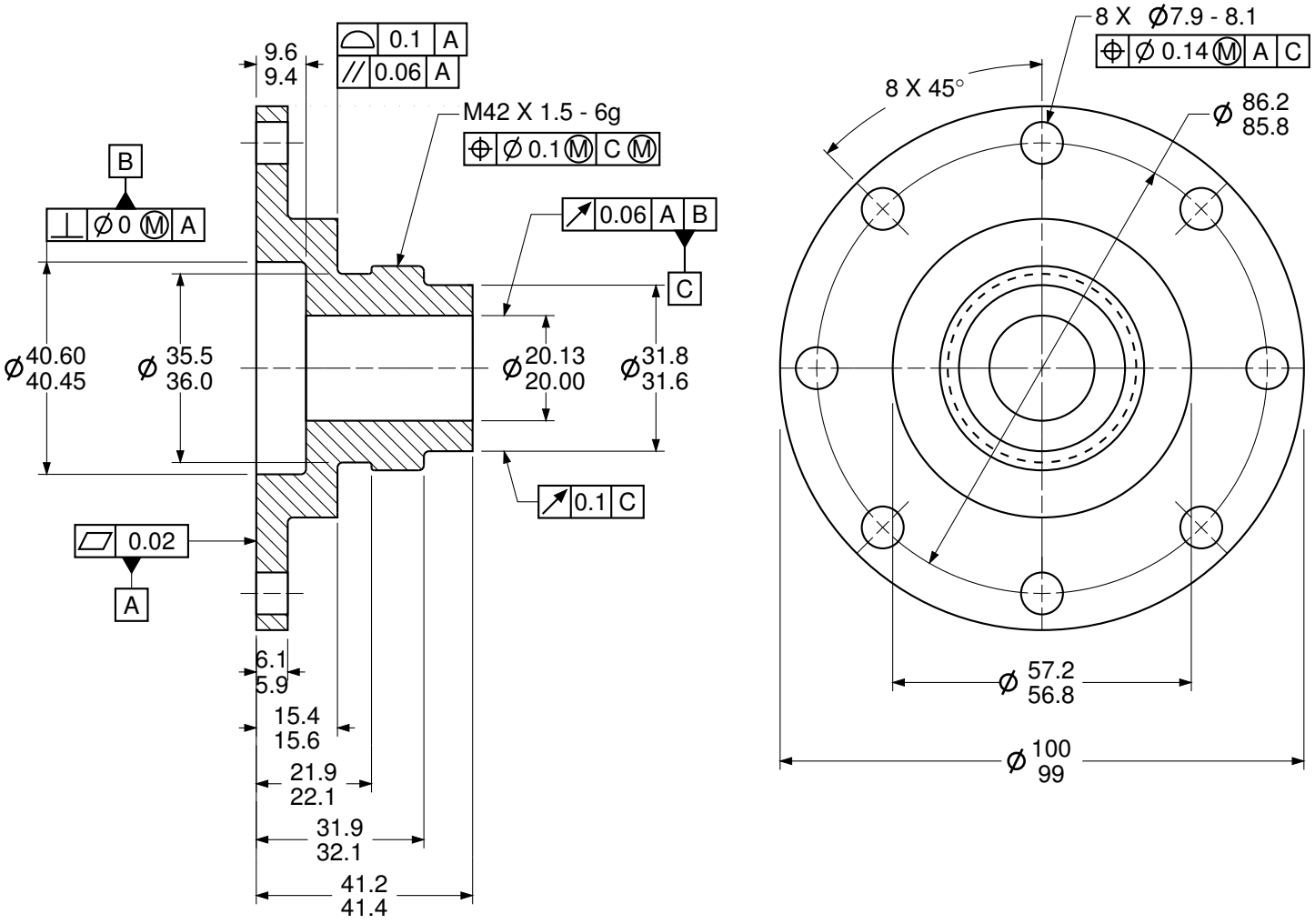


Figure 111: Dimensioning with tolerances `[Plate.m4]`.

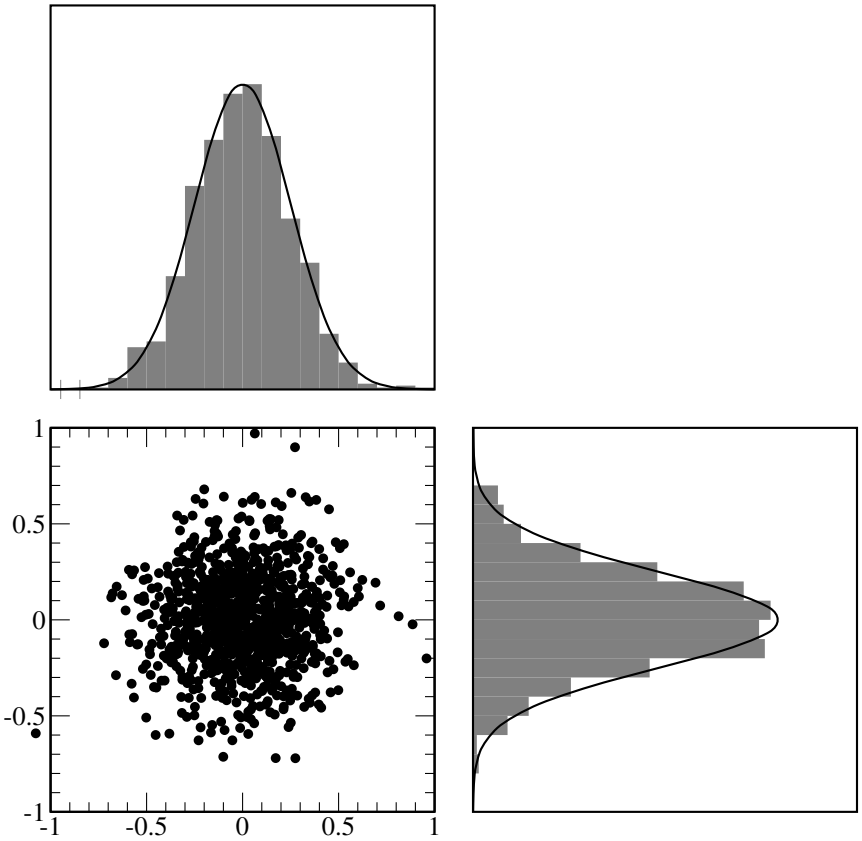


Figure 112: Testing random number generation using dpic macro `randn(`*array name, n, mean, std dev*`)` which calls pic built-in `rand() [random.m4]`.

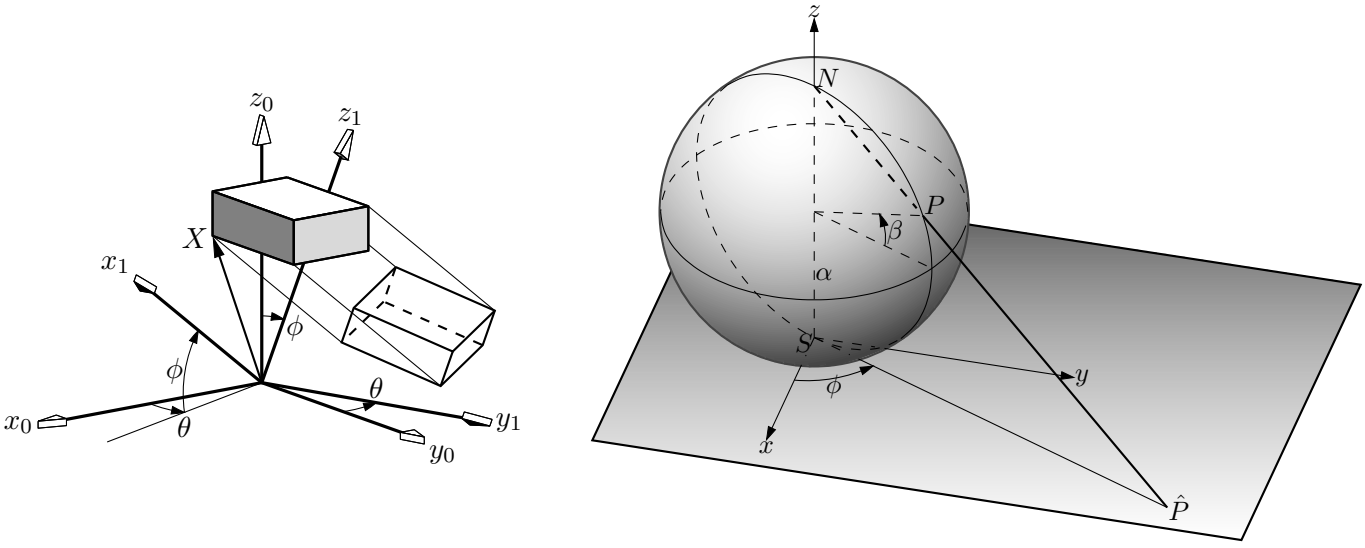


Figure 113: Test of `project` and other `lib3D` macros, showing the projection of a solid onto the y_1, z_1 plane by sighting along the x_1 axis. `[exp.m4]`.

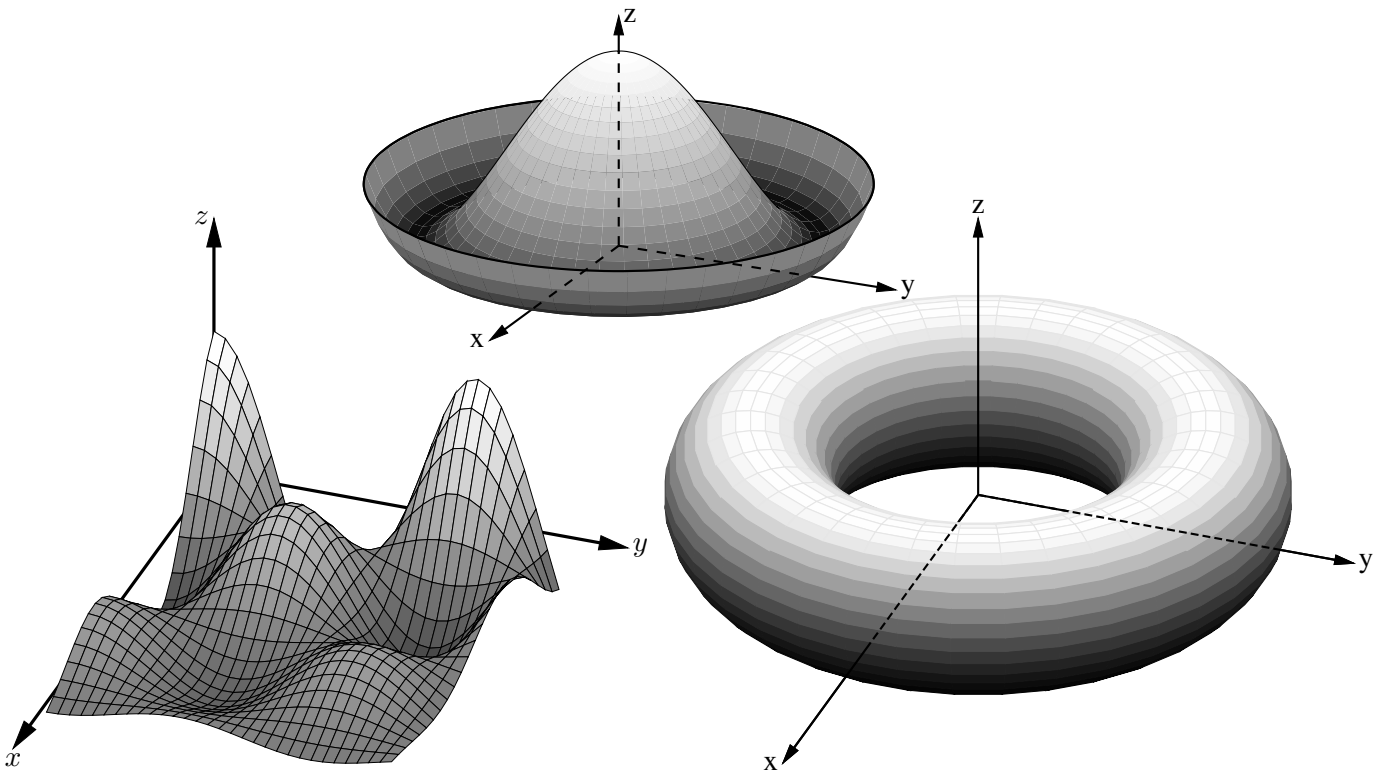


Figure 114: Plotting surfaces using gray scales `[graysurf.m4]`.

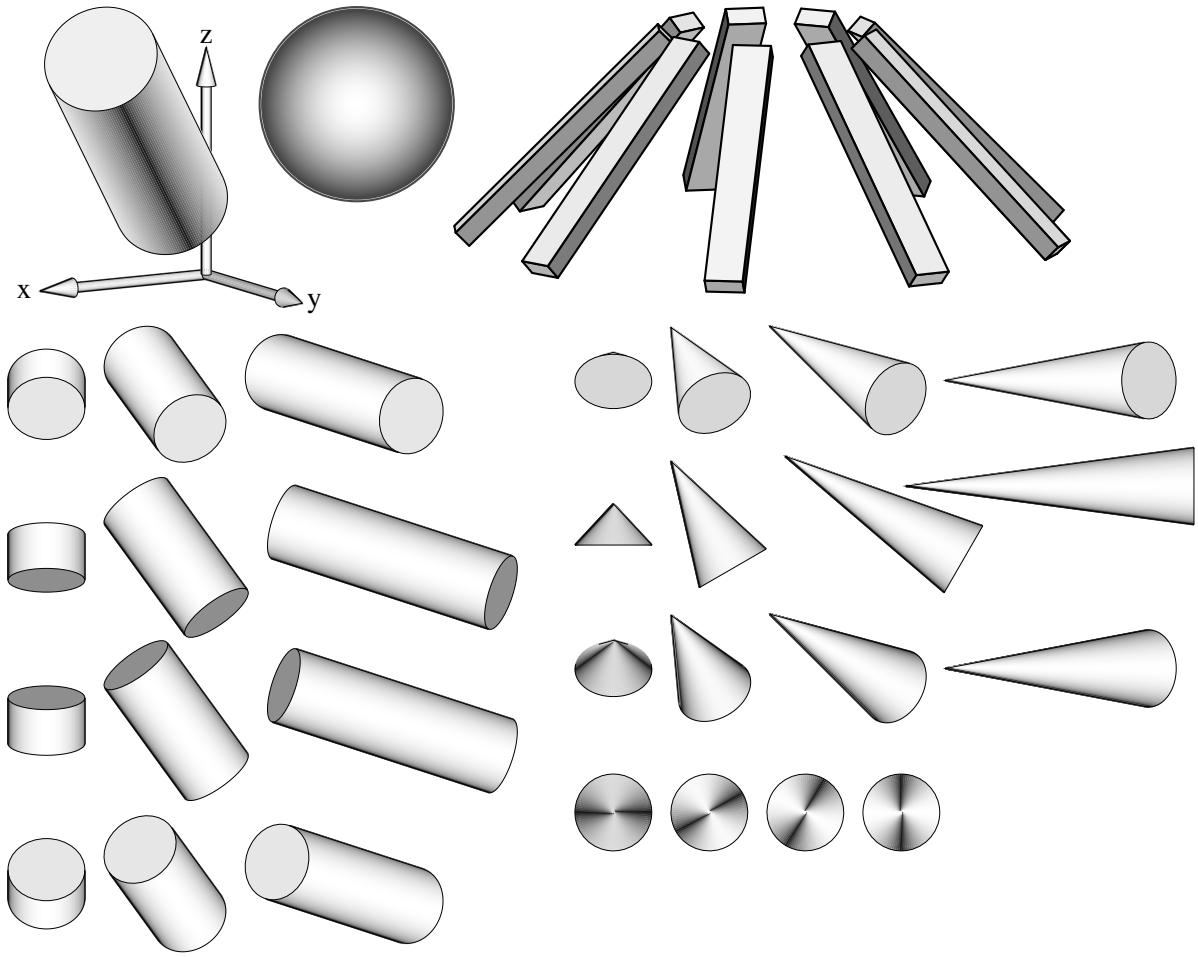


Figure 115: Basic shapes `[shapes.m4]`.

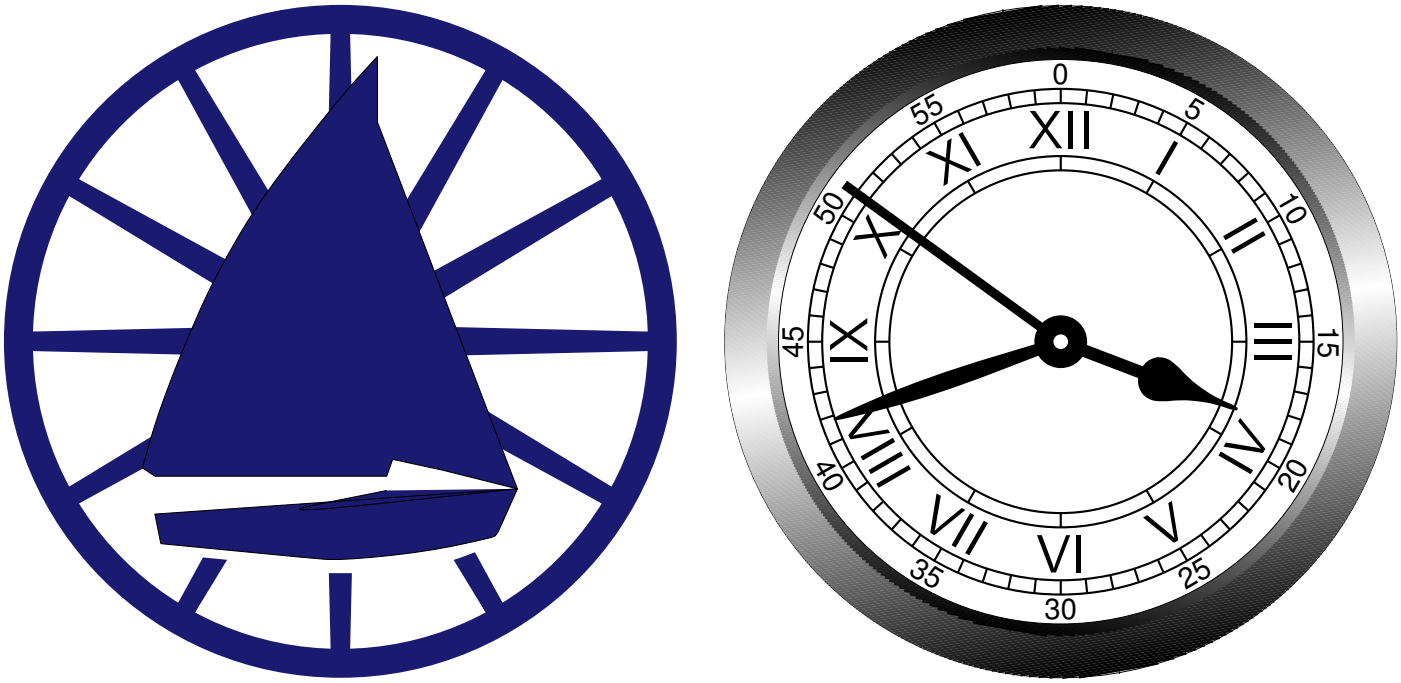


Figure 116: Conestoga Sailing Club (illustrating the filling of arbitrary shapes) and an antique clock face with shading and rotated text `[csc.m4]`.

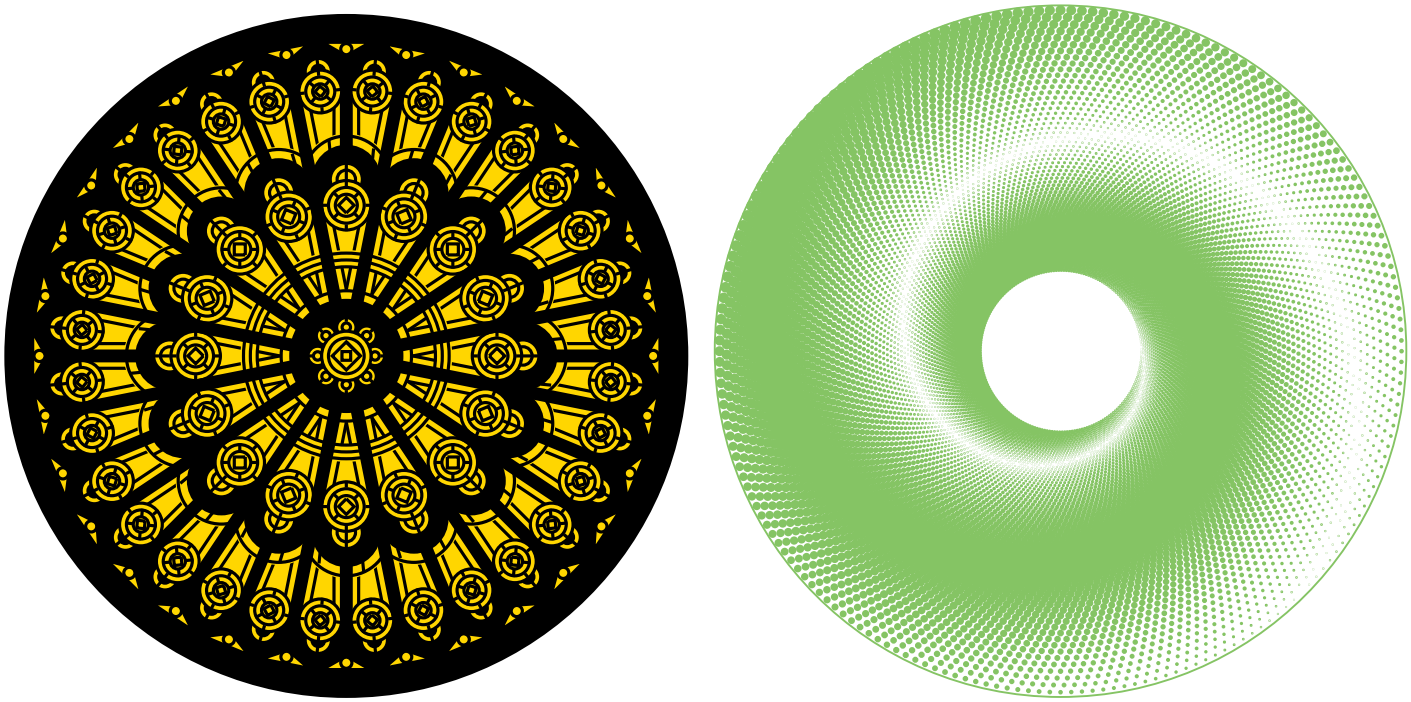


Figure 117: The left object, used for testing `dpic,` is redrawn from a detail of the set design for the musical *Dracula.* This consumes much LaTeX main memory but can be produced directly as pdf using `dpic -d`, as svg using `dpic -v`, or as postscript using `dpic -r` since no text formatting is required. The right object adjusts the size of dots to produce a halftone effect `[rose.m4]`.

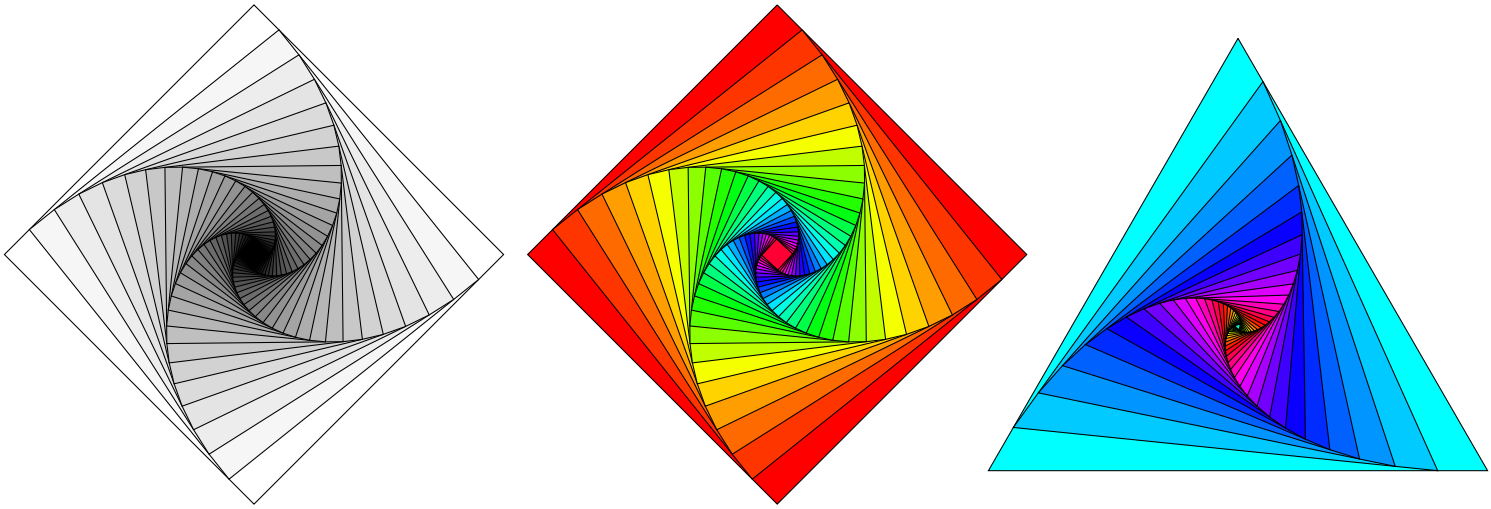


Figure 118: Variations on M. Goossens, S. Rahtz, and F. Mittelbach, *The LaTeX Graphics Companion,* Addison-Wesley 1997, pp. 57-58 `[diamond.m4]`.

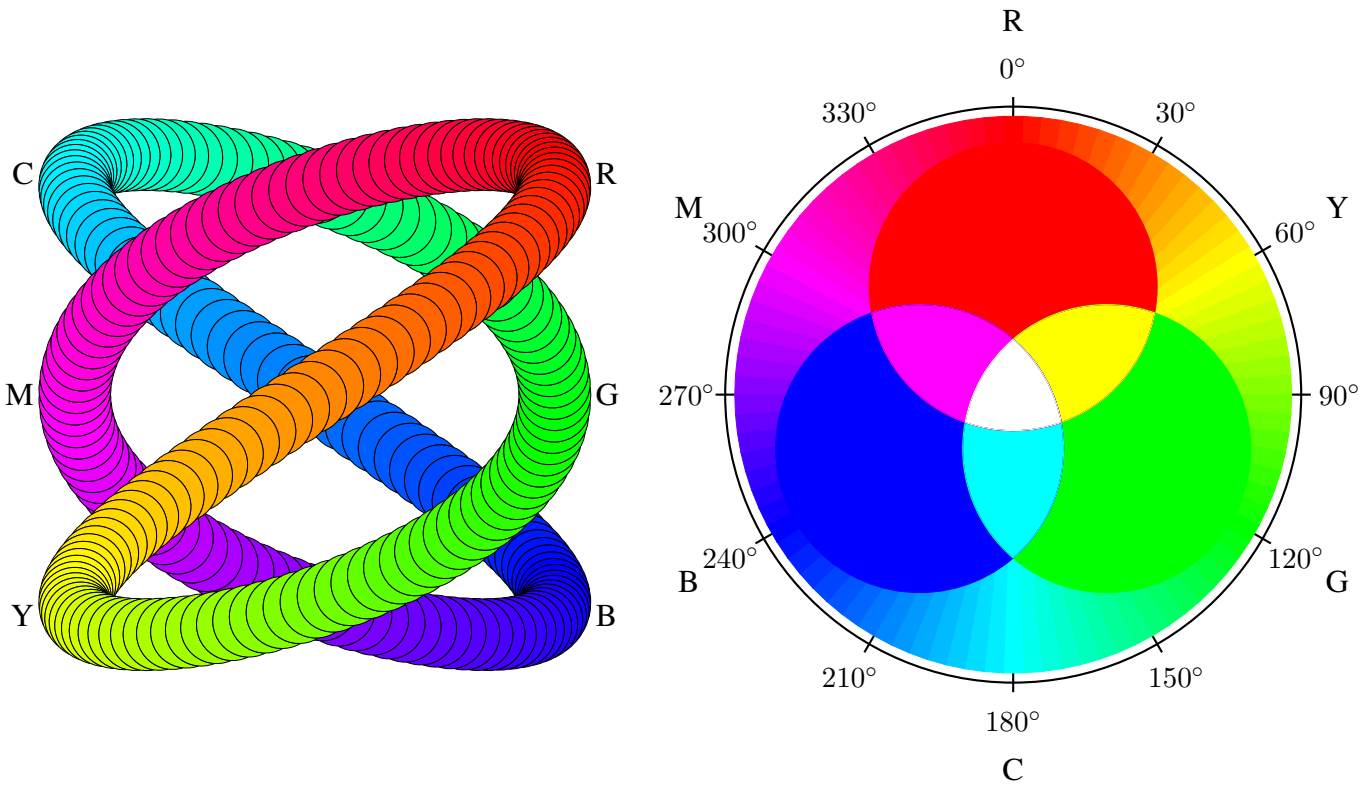


Figure 119: An exercise in calculating RGB colours `[worm.m4]`.

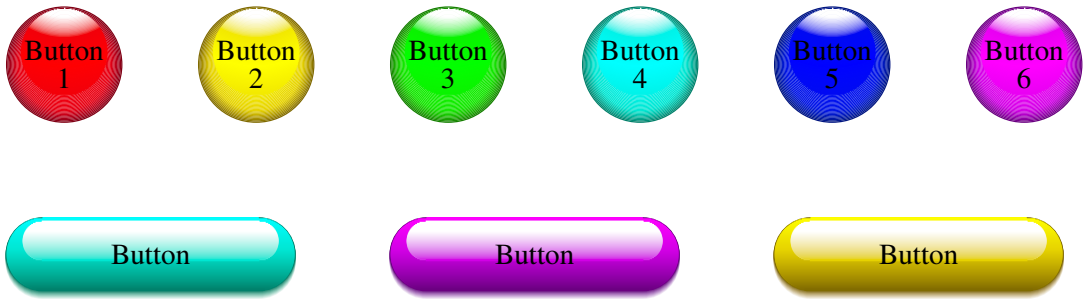


Figure 120: Shading in color `[Buttons.m4]`.

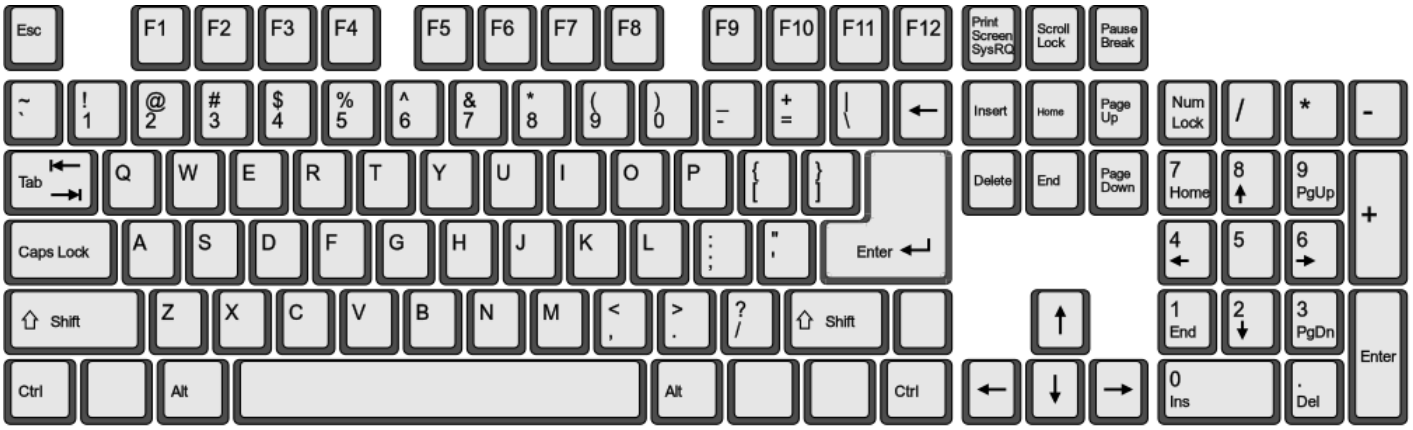


Figure 121: This diagram has been produced as svg with dpic -v (then converted to pdf for inclusion in examples.pdf) `[keyboard.m4]`.

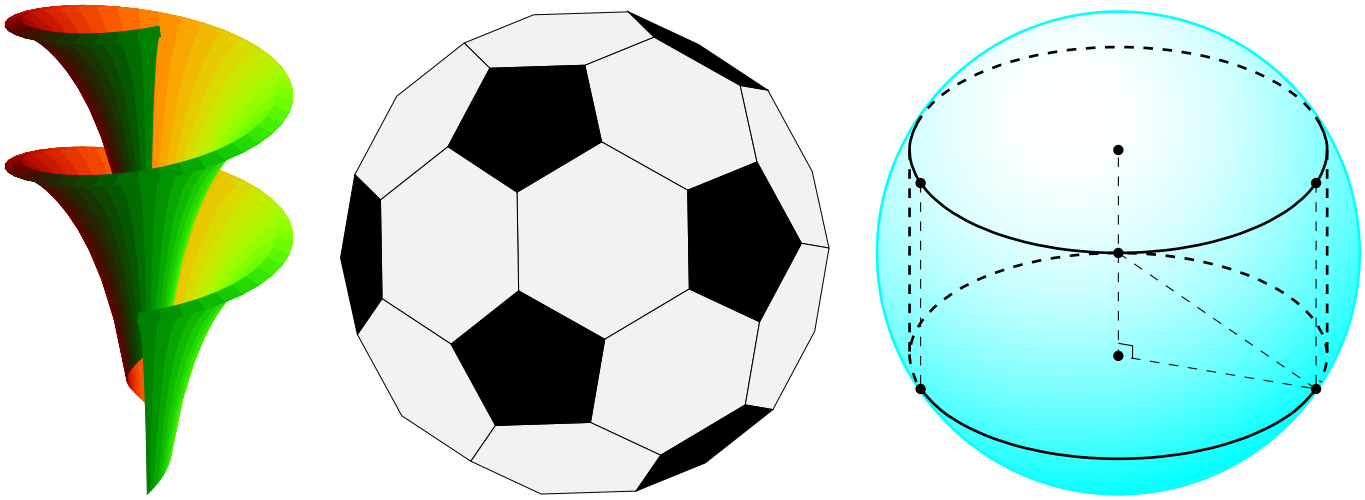


Figure 122: Dini surface, an icosahedron, and a sphere with inscribed cylinder `[Dini.m4]`.

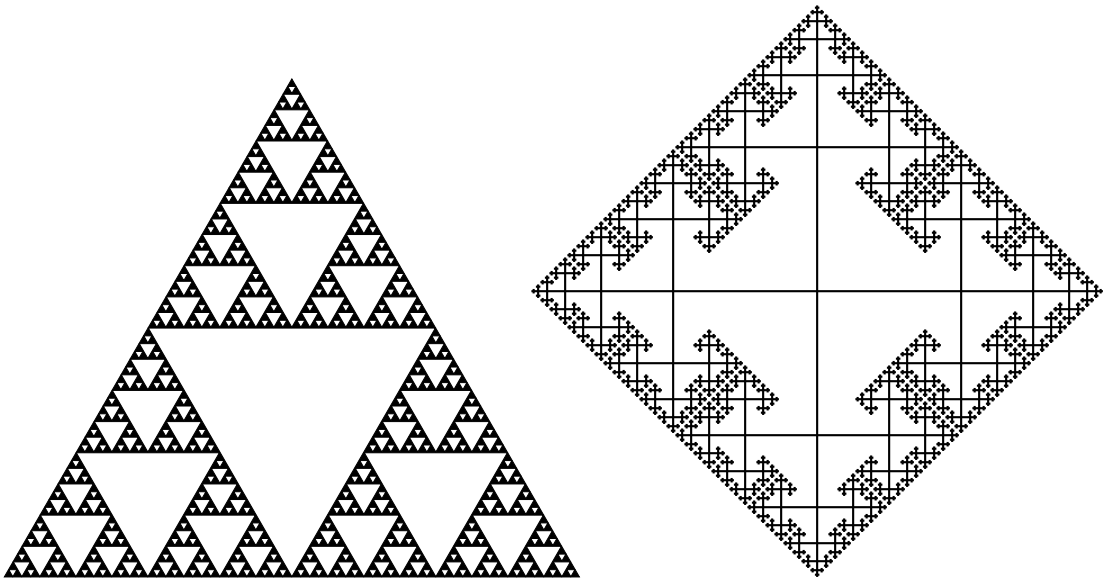


Figure 123: The Sierpinski triangle and a Cayley graph: tests of pic macro recursion `[Sierpinski.m4]`.

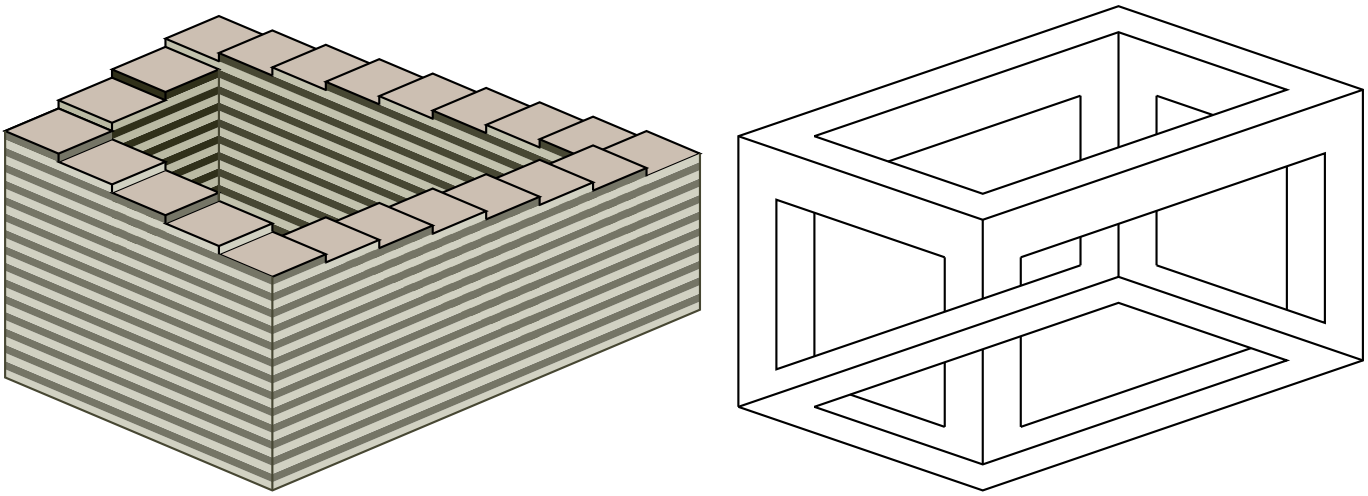


Figure 124: Penrose stairs and an Escher-like object `[Escher.m4]`.

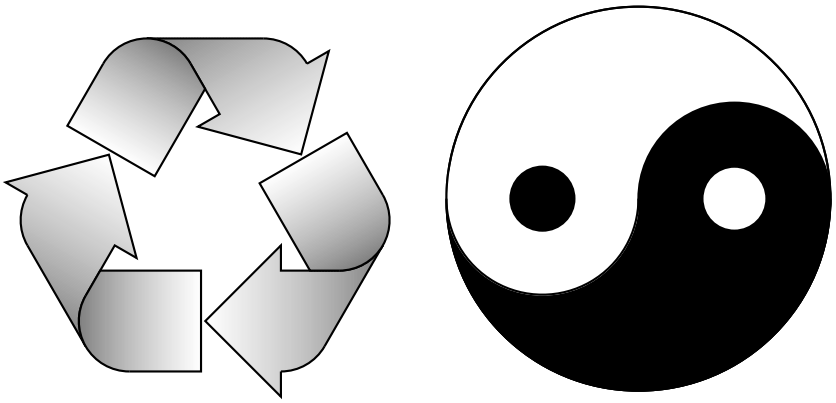


Figure 125: Modest repetition and partial fill `[recycle.m4]`.

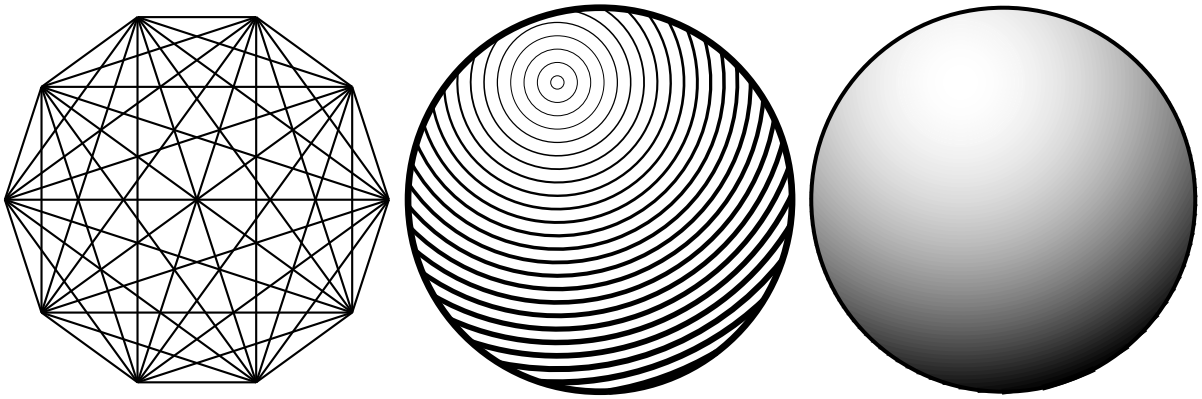


Figure 126: Simple diagrams that are easily drawn by looping `[ex15.m4]`.

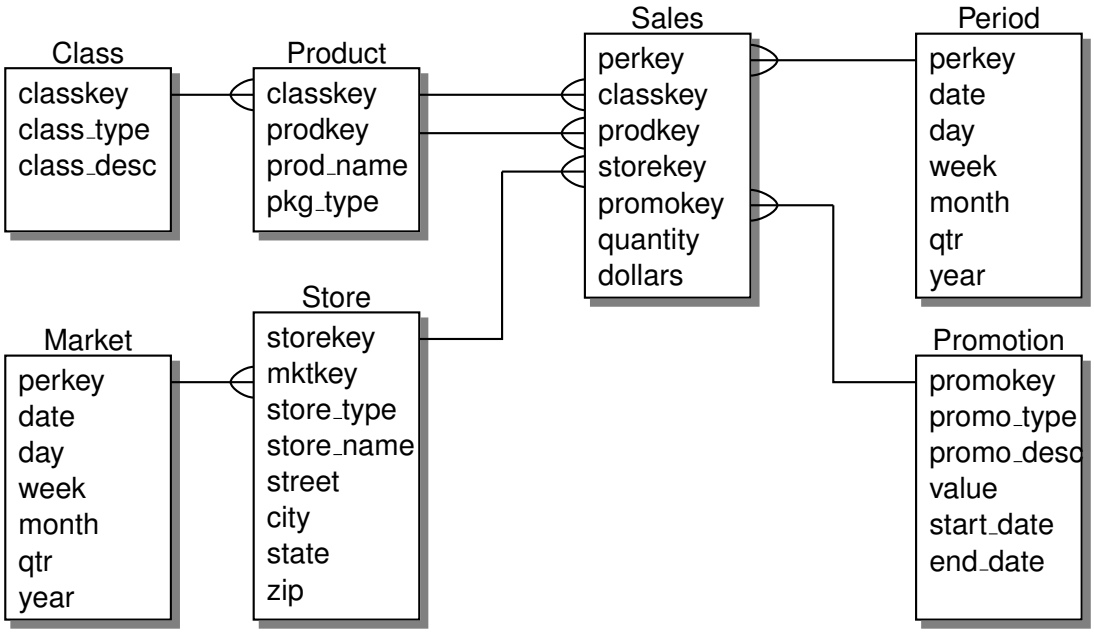


Figure 127: Illustrating `shadebox` and a custom crowfoot line termination `[Crow.m4]`.



Figure 128: Overlaying a figure with line graphics `[Incleps.m4]`.