Examples: Version 10.1.4

This is a collection of diagrams the author has had occasion to produce using m4 circuit macros and others, and dpic or gpic. In some cases there are 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 is achieved by sticking to one pic interpreter and one postprocessor.

This document duplicates a few diagrams from the manual Circuit_macros.pdf. There are also a few files in the examples directory that are not included in this document. To process *file*.m4, for example, type "make *file*.pdf".

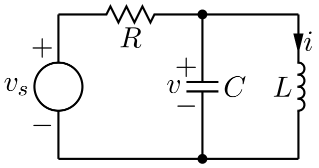


Figure 1: The quick-start example from the manual [quick.m4].

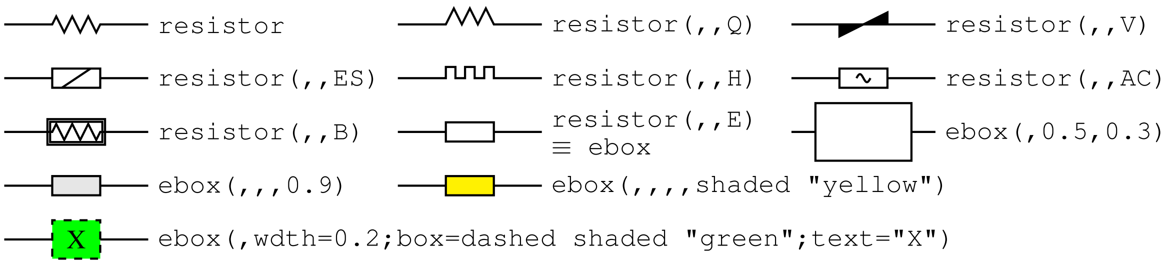


Figure 2: Resistors, showing some variations and the ebox [Resistors.m4].

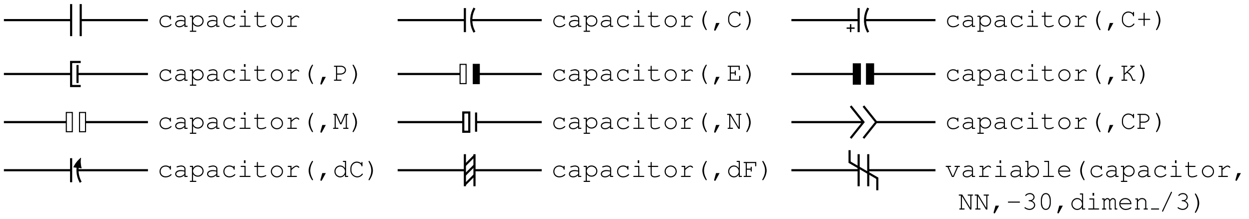


Figure 3: Capacitors [Capacitors.m4].

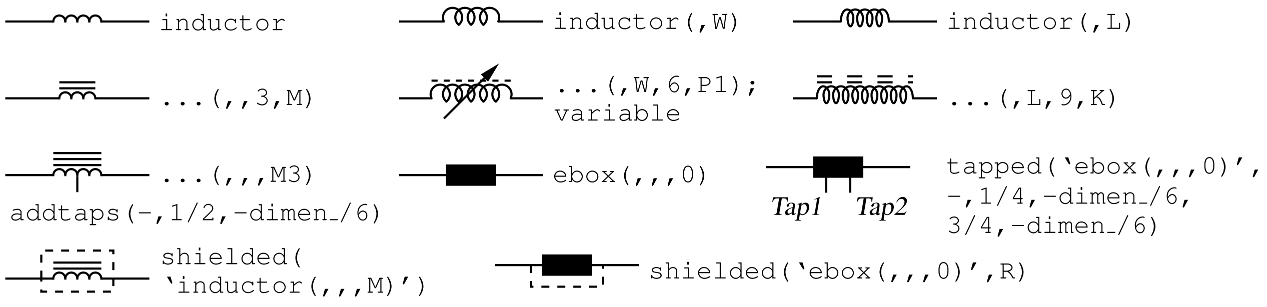


Figure 4: Inductors [Inductors.m4].

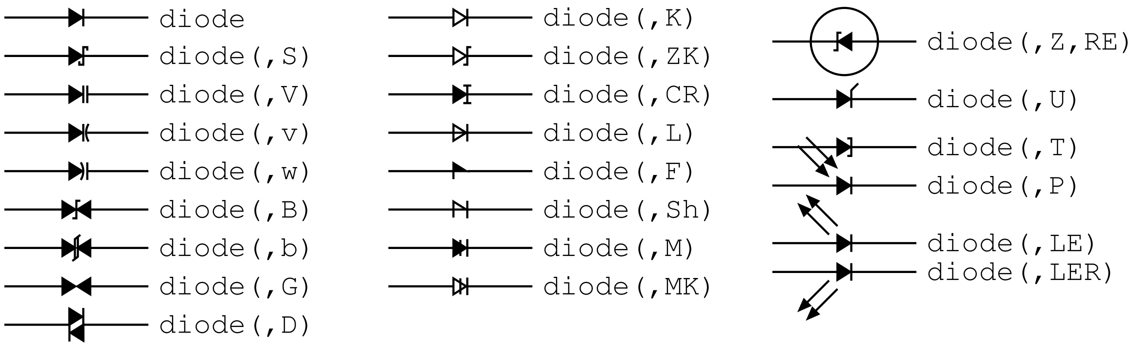


Figure 5: Diodes: appending a K to the second argument draws an open arrowhead [Diodes.m4].

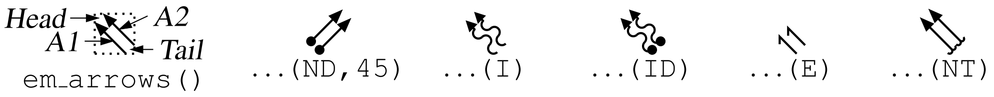


Figure 6: Radiation arrows [Emarrows.m4].

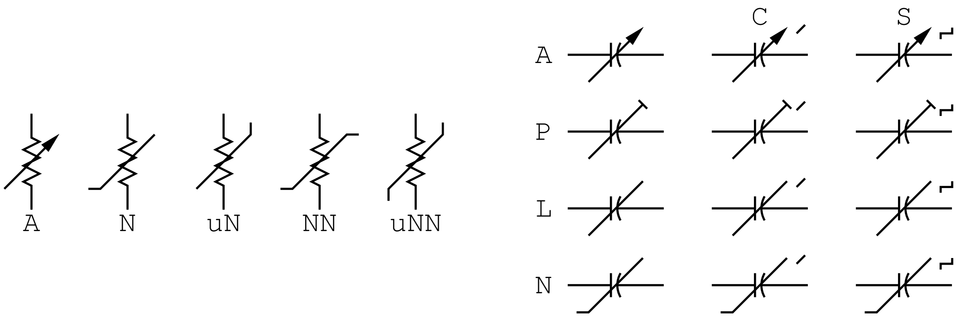


Figure 7: Arrows and marks for showing variability [Variable.m4].

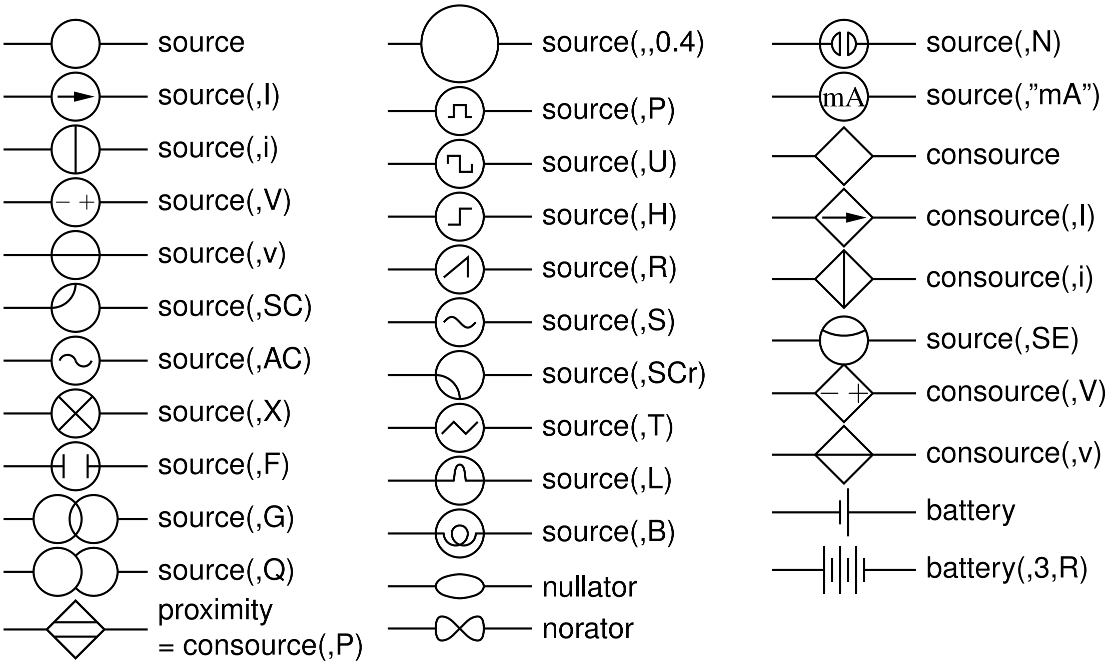


Figure 8: Sources and source-like elements `[Sources.m4]`.

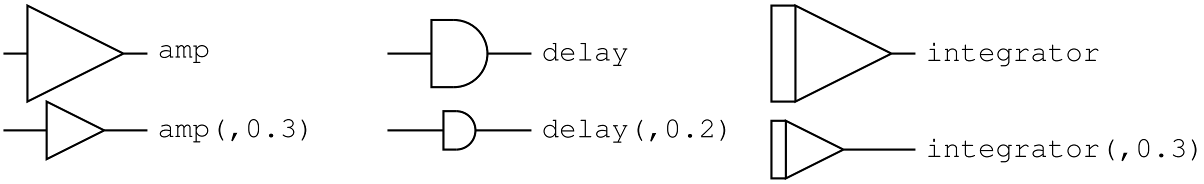


Figure 9: Macros `amp`, `delay`, and `integrator` `[AmpTable.m4]`.

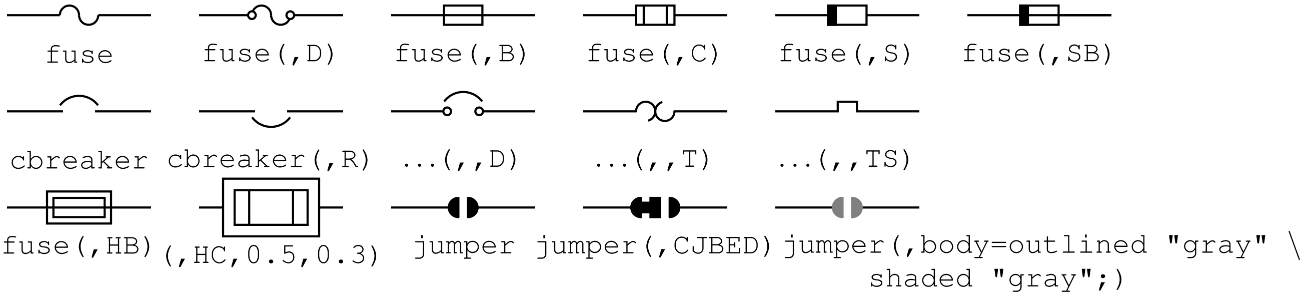


Figure 10: Macros `fuse` and `cbreaker` `[Fuses.m4]`.

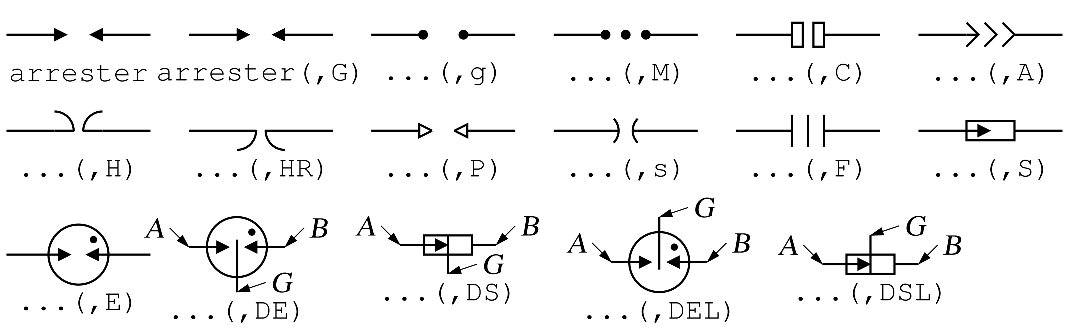


Figure 11: The `arrester` macro `[Arresters.m4]`.

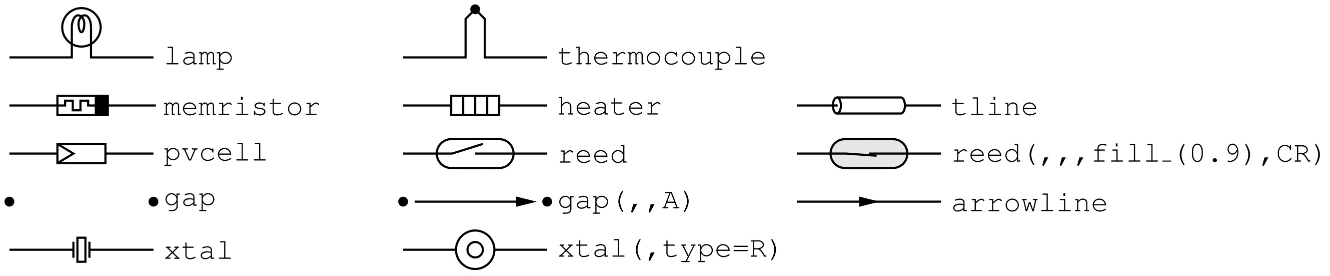


Figure 12: Additional two-terminal elements `[MoreTable.m4]`.

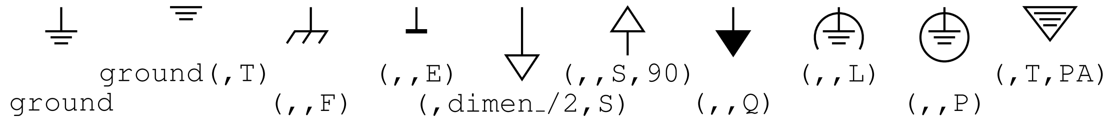


Figure 13: Ground symbols `[Grounds.m4]`.

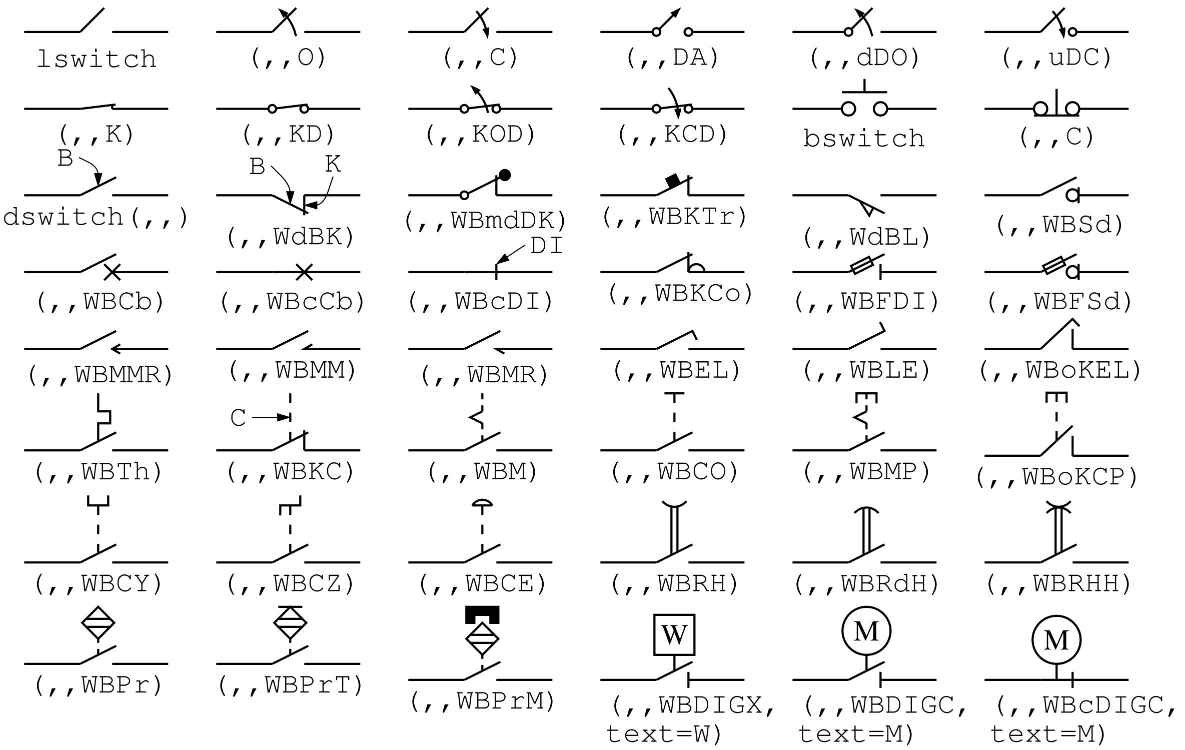


Figure 14: The switch macros; switch(,,,L|B|D) is a wrapper for lswitch, bswitch, and dswitch [Switches.m4].

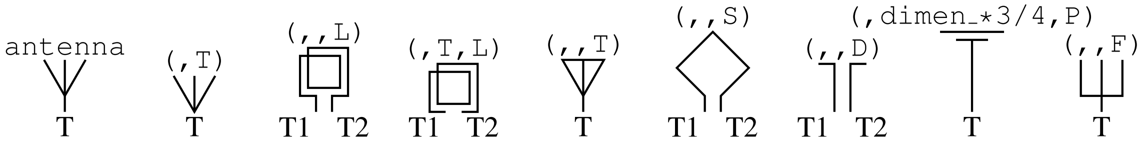


Figure 15: Antenna symbols [Antennas.m4].

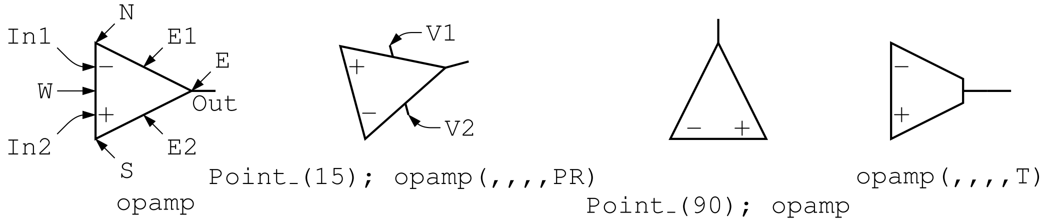


Figure 16: The opamp [Opamp.m4].

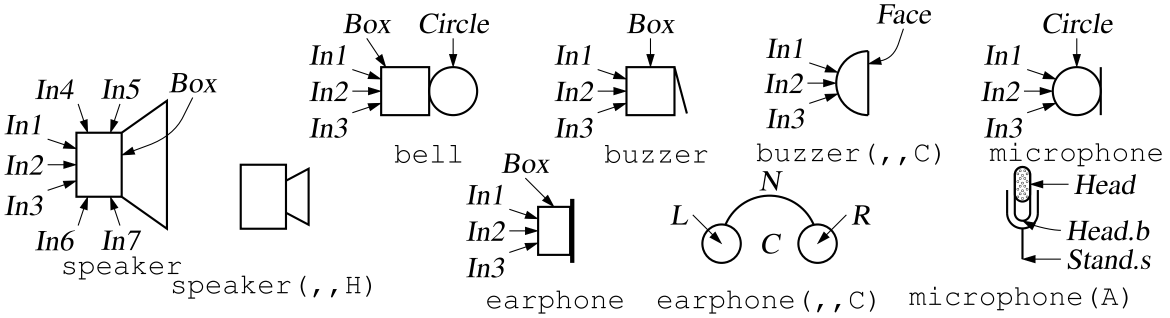


Figure 17: Audio elements [Audio.m4].

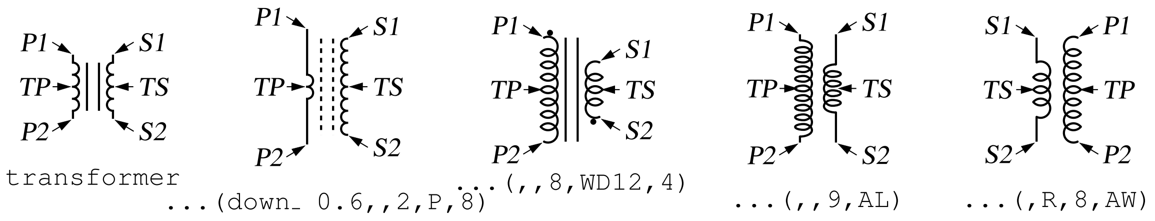


Figure 18: Some variations of the transformer element, drawing direction down [Xform.m4].

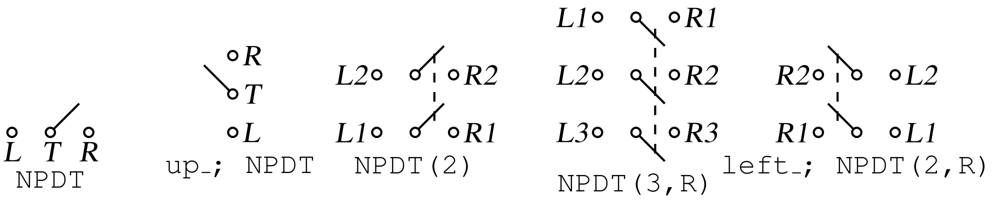


Figure 19: Double throw with the NPDT macro [NPDT.m4].

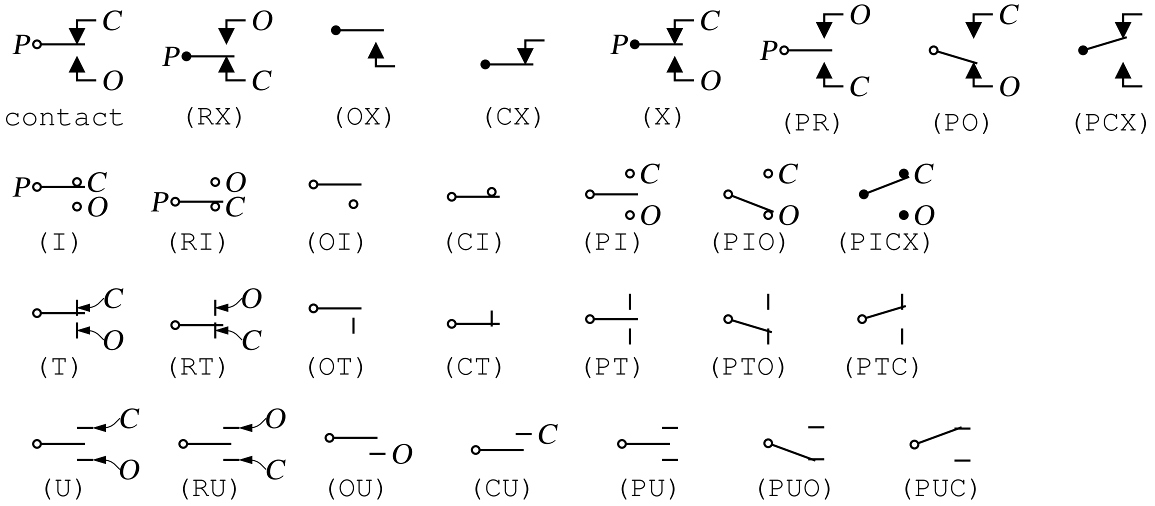


Figure 20: A non-exhaustive sampling of contact macro variations [Contact.m4].

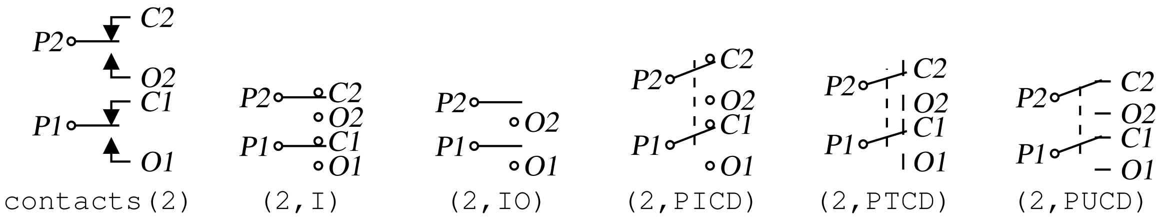


Figure 21: The `contacts` macro `[Contacts.m4]`.

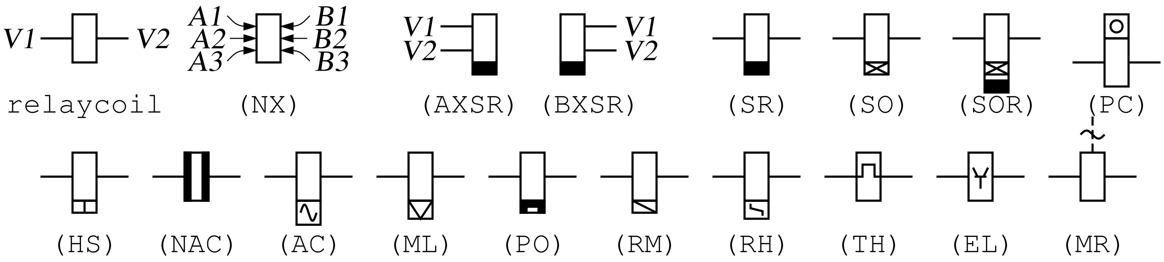


Figure 22: The `relaycoil` macro `[relaycoil.m4]`.

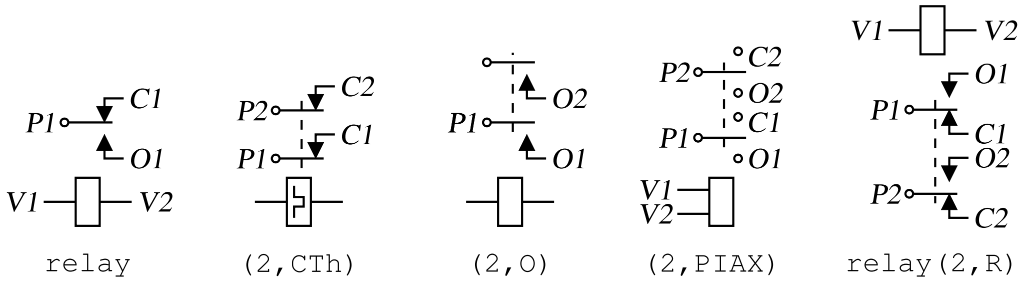


Figure 23: Some variants of `relay` `[Relay.m4]`.

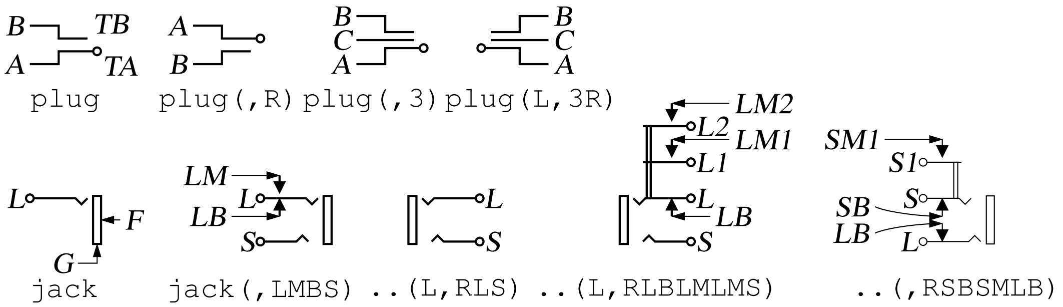


Figure 24: The `jack` and `plug` macros `[Jack.m4]`.

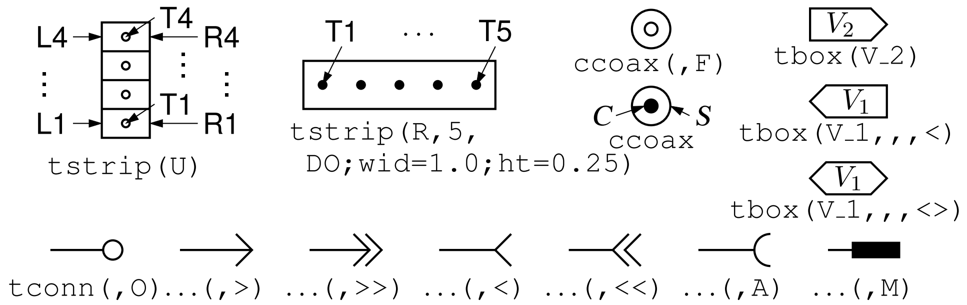


Figure 25: The `tstrip`, `ccoax`, `tconn`, and `tbox` macros `[Conn.m4]`.

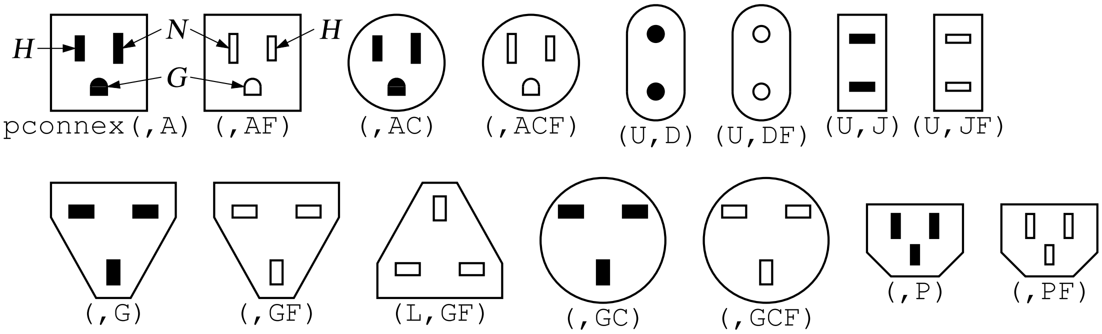


Figure 26: The `pconnex` macro `[Pconn.m4]`.

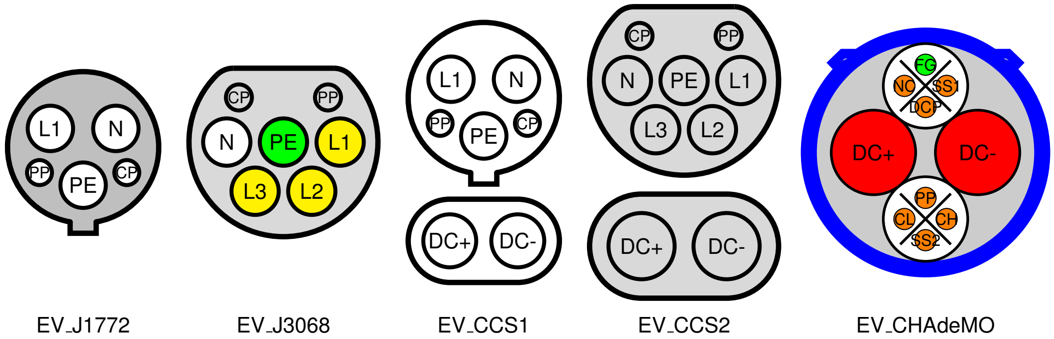


Figure 27: Electric vehicle charging plug patterns make extensive use of *key=value* pairs to set options `[EVplugs.m4]`.

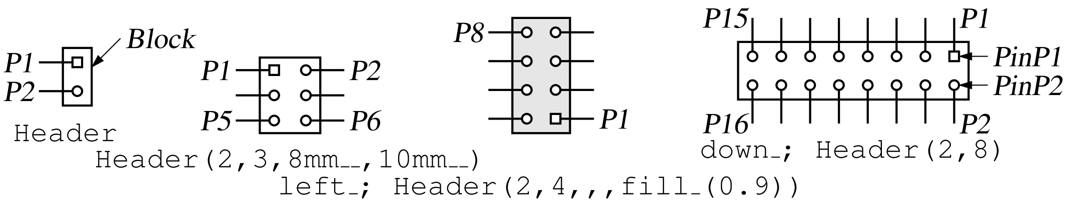


Figure 28: The `Header` macro `[Headers.m4]`.

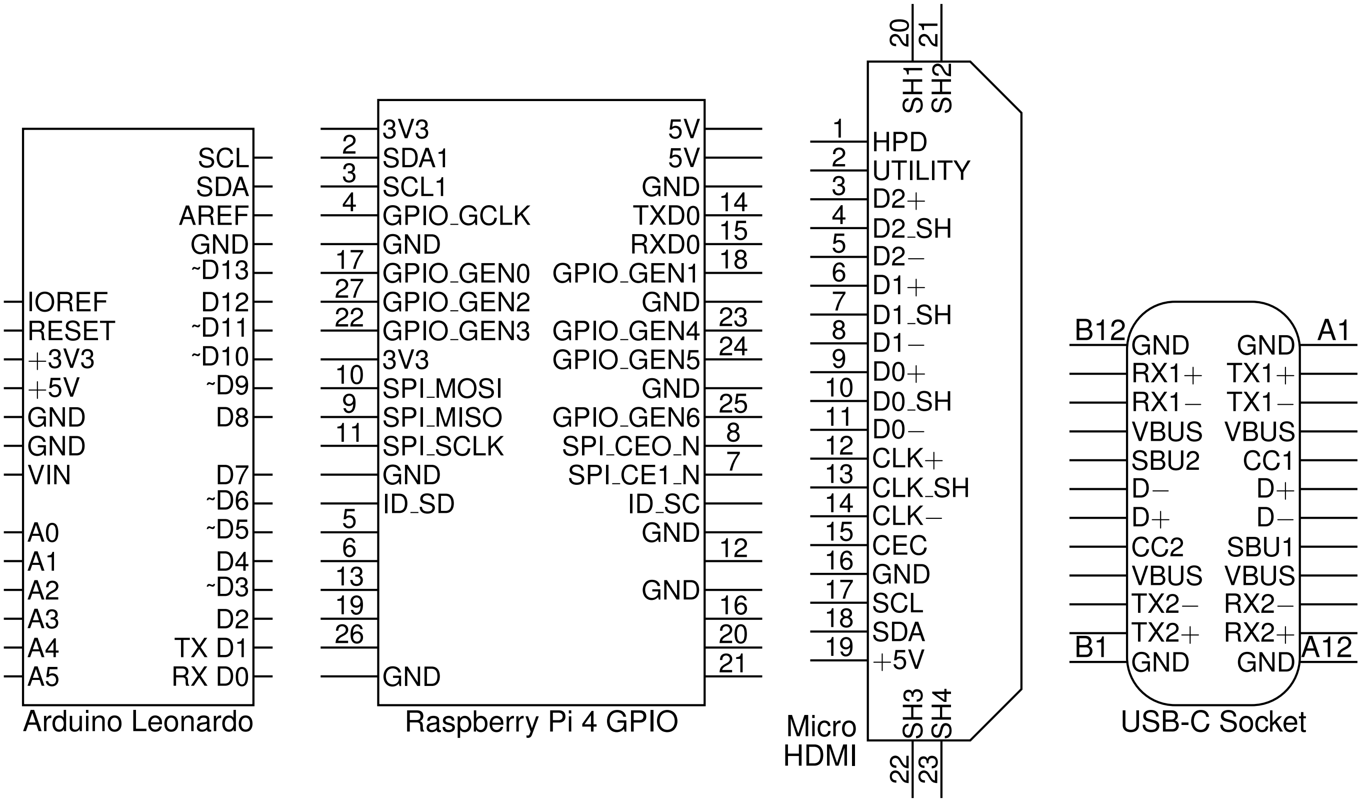


Figure 29: Some connectors with simple geometry and lists of labels `[Connectors.m4]`.

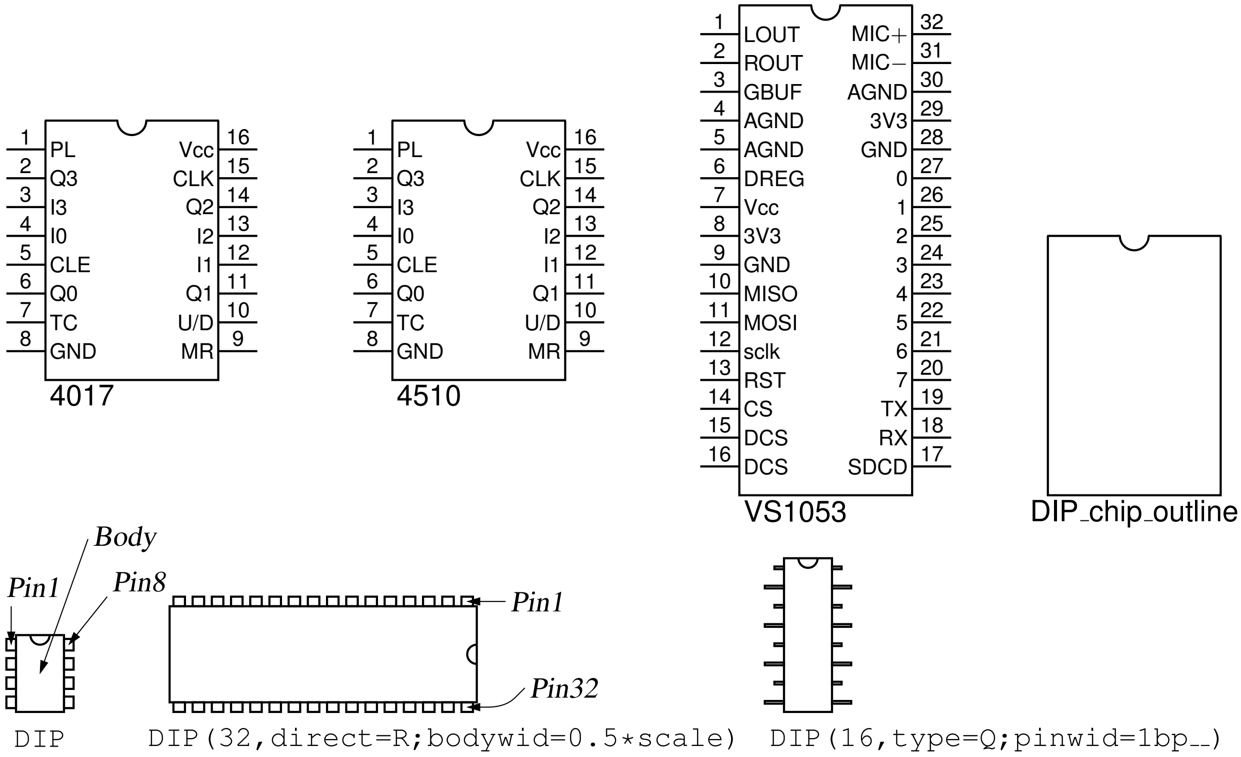


Figure 30: IC package outlines and examples `[Chips.m4]`.

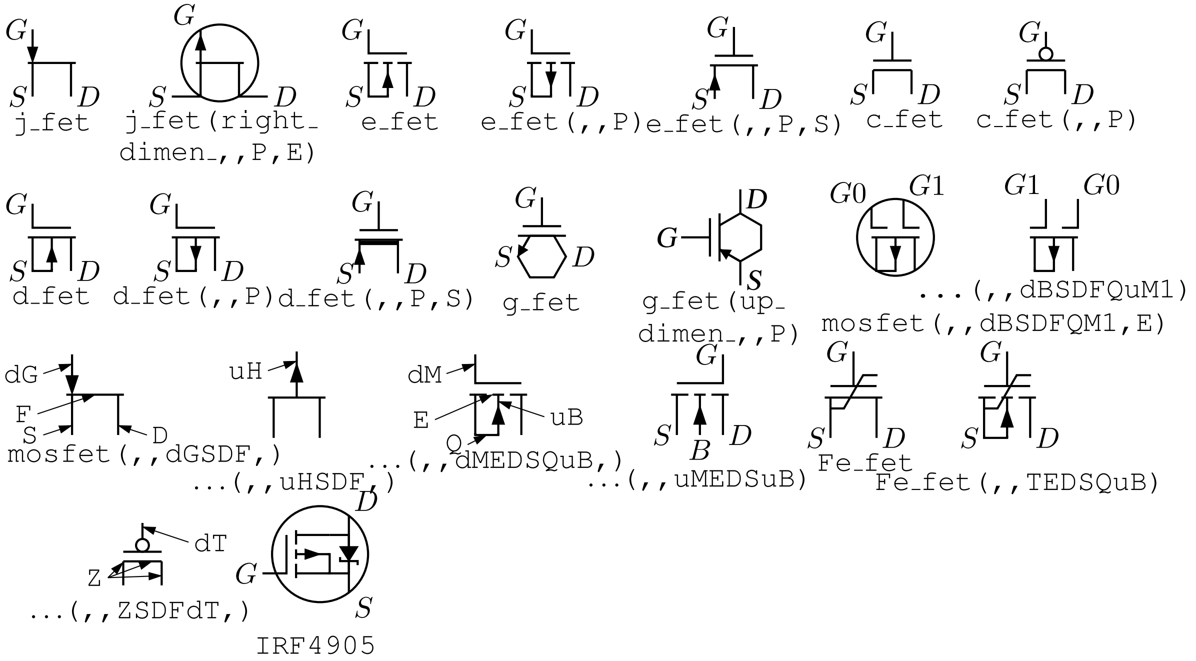


Figure 31: FETs, showing programmable components and example customizations [fet.m4].

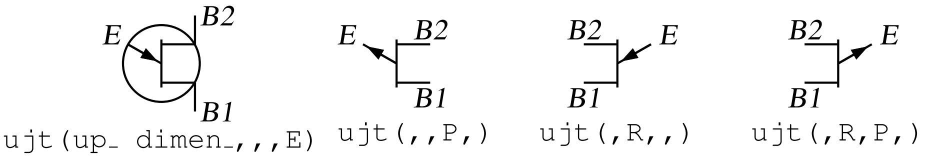


Figure 32: UJT examples [ujt.m4].

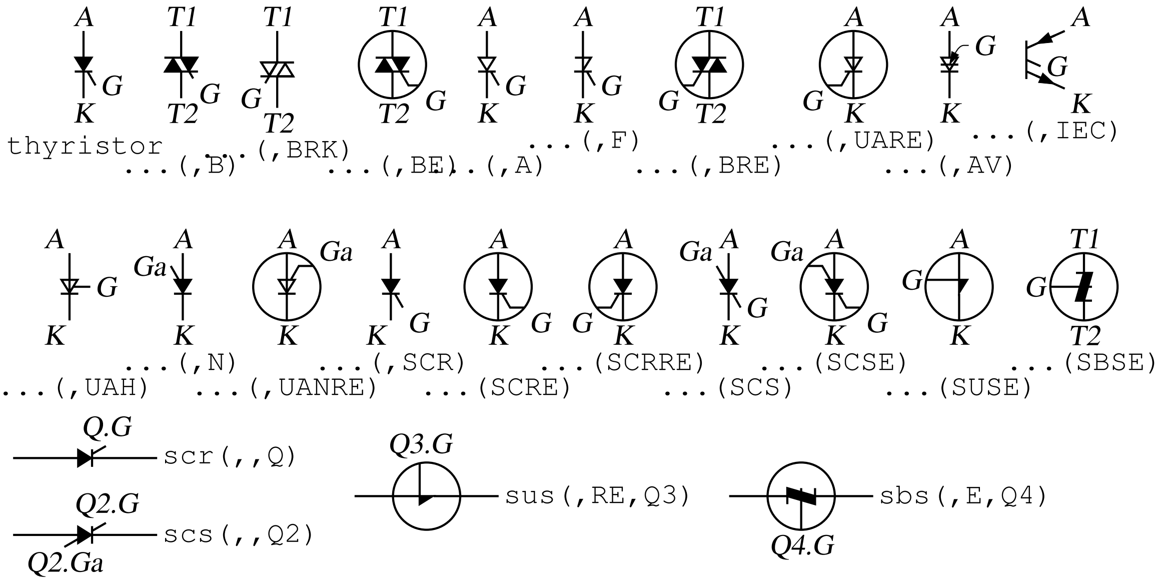


Figure 33: Thyristor examples. The thyristor is a 3- or 4-terminal composite element [thyristor.m4].

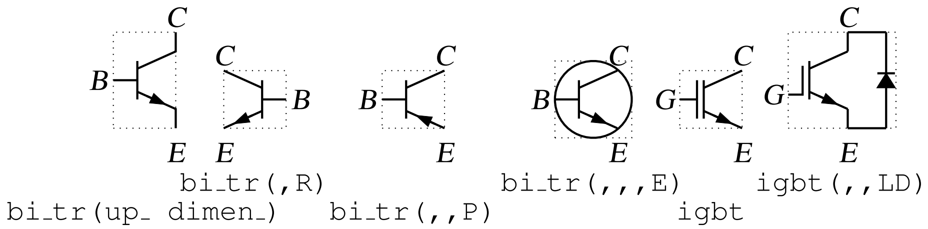


Figure 34: Bipolar transistors (drawing direction: up) [Bip.m4].

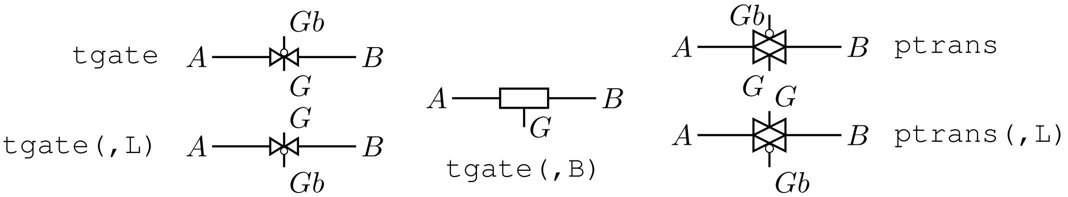


Figure 35: The tgate and ptrans elements [Tgate.m4].

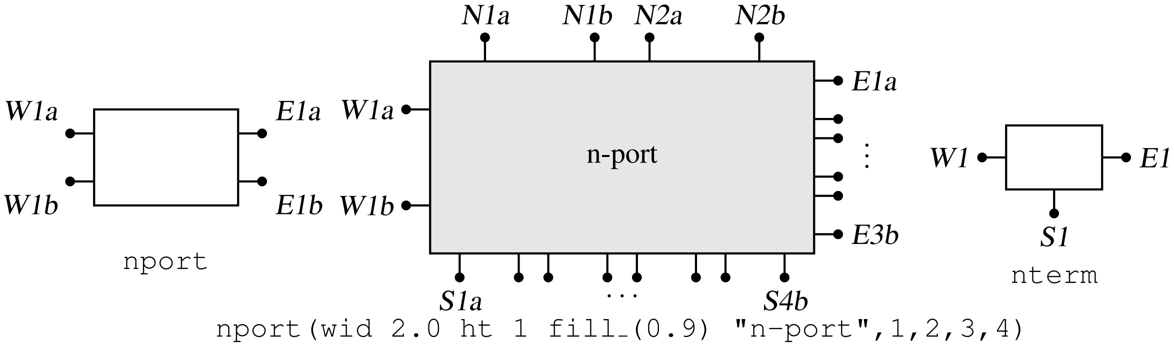


Figure 36: The nport and nterm macros [Nport.m4].

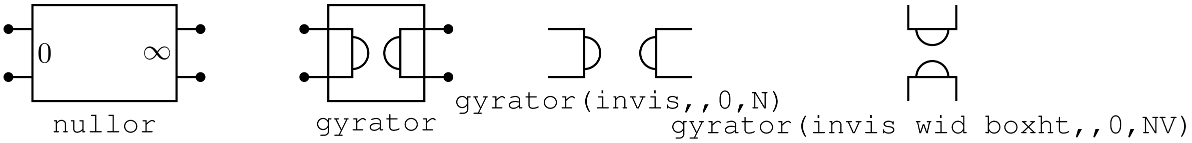


Figure 37: Some customizations of nport [NLG.m4].

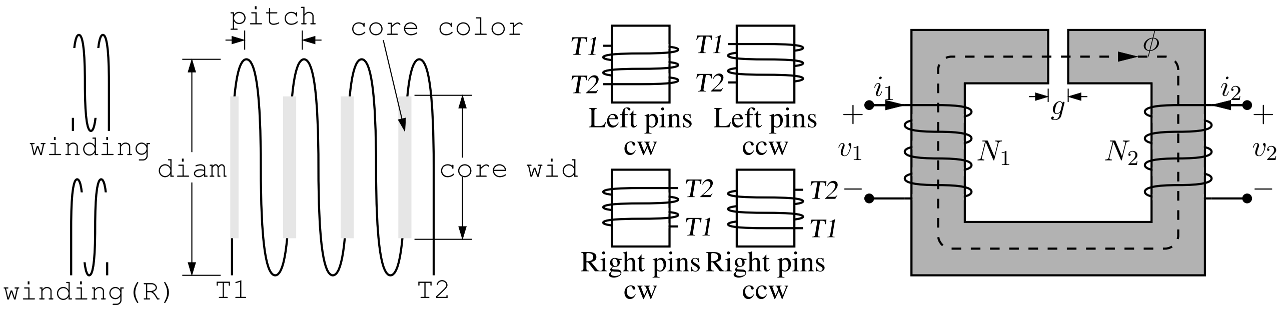


Figure 38: The macro `winding(L|R,diam,pitch,turns,core wid,core color) [Windings.m4]`.

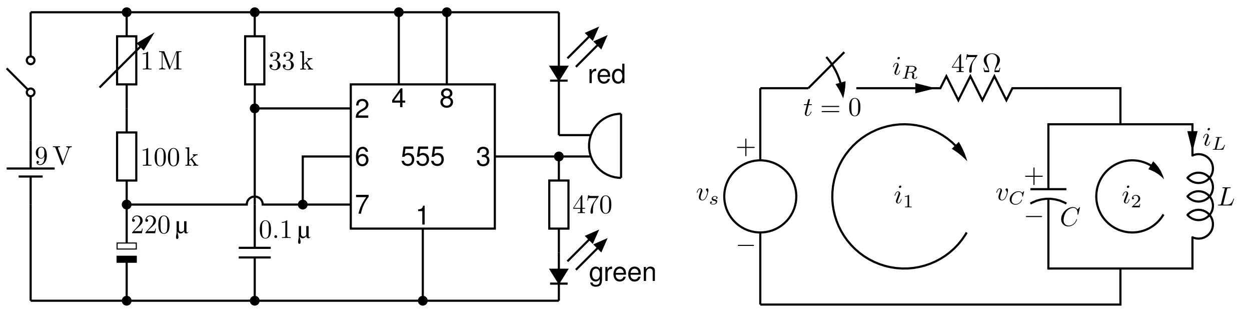


Figure 39: Two simple labeled circuits `[ex01.m4]`.

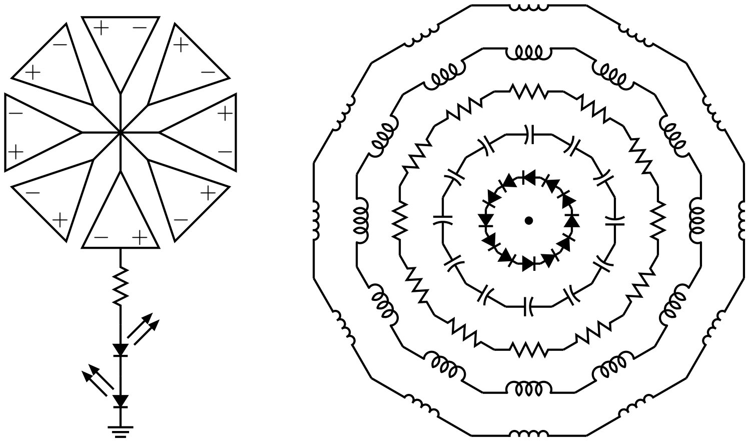


Figure 40: Elements at obtuse angles `[ex02.m4]`.

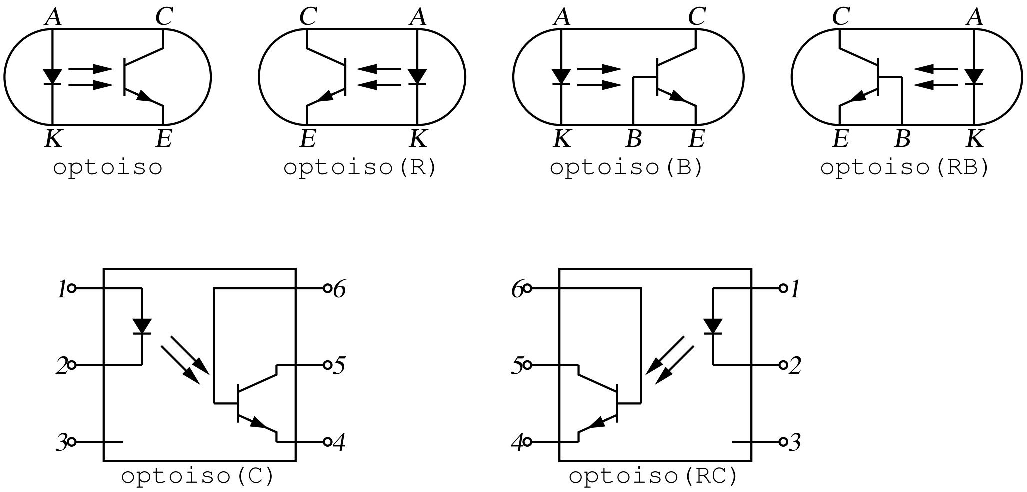


Figure 41: Optical isolator: a circuit with right or left orientation [Optoiso.m4].

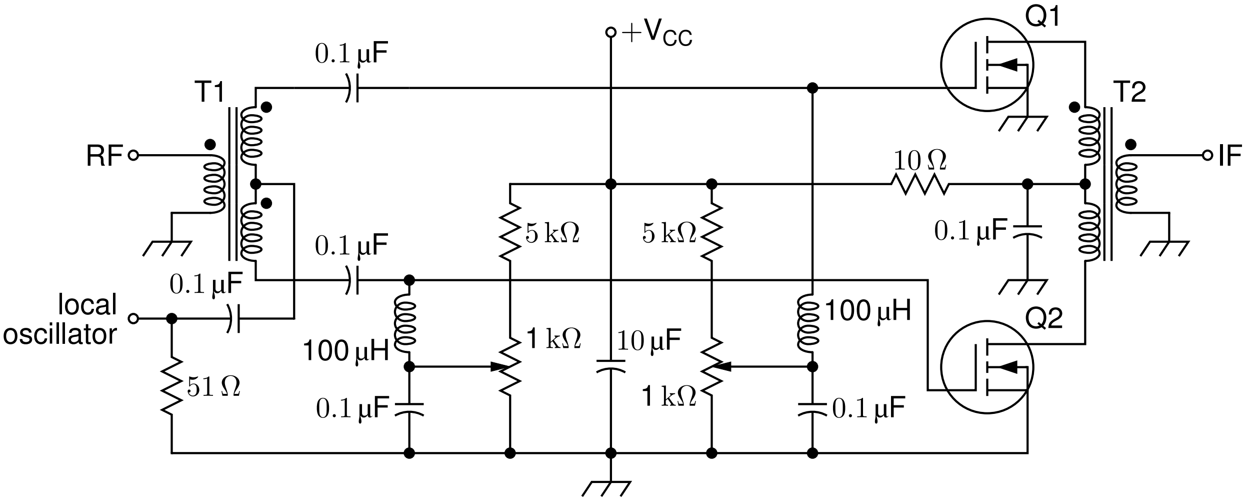


Figure 42: A balanced mixer, using mosfet and a custom transformer [Mixer.m4].

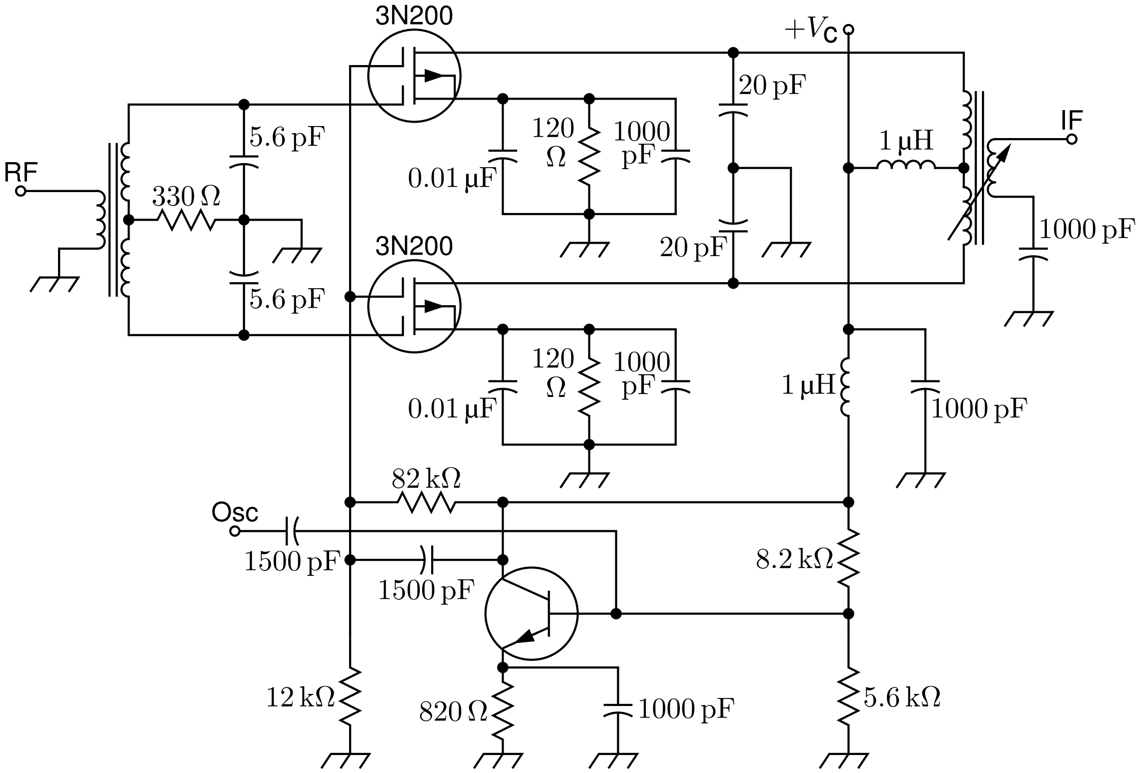


Figure 43: A push-pull mixer, showing FETs with multiple gates [PushPull.m4].

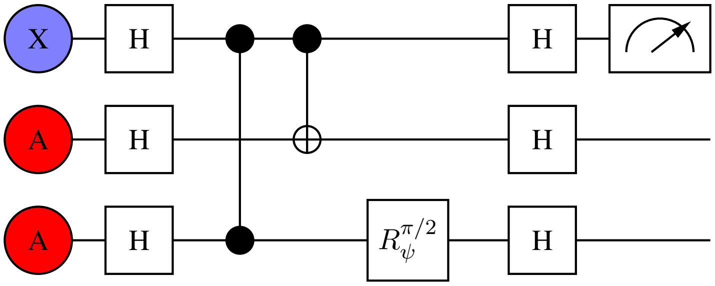


Figure 44: A quantum circuit [Quantum.m4].

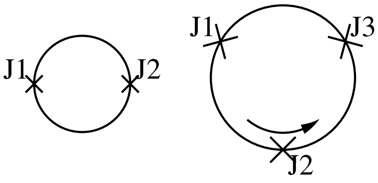


Figure 45: Superconducting quantum interface device (drawing direction down) [SQUID.m4].

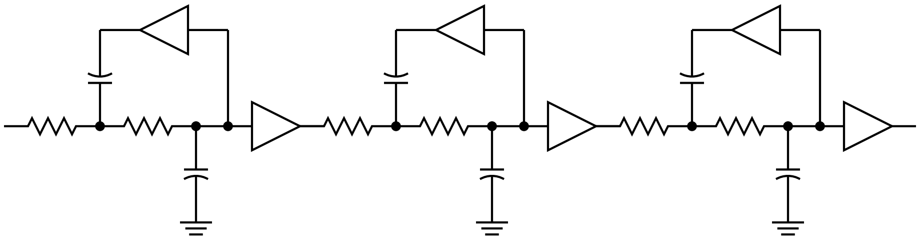


Figure 46: A six-pole filter [Sixpole.m4].

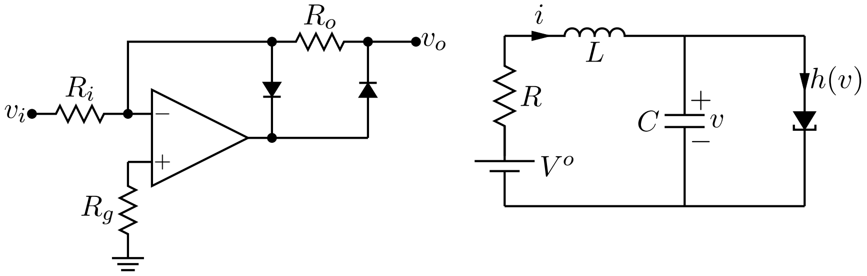


Figure 47: Precision half-wave rectifier and a tunnel diode circuit (illustrating opamp, diode, resistor, ground, and labels) [ex18.m4].

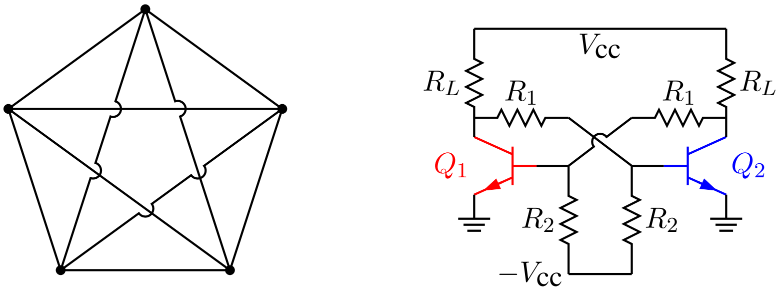


Figure 48: Non-planar graph and bistable circuit (illustrating the crossover macro and colored elements) [ex10.m4].

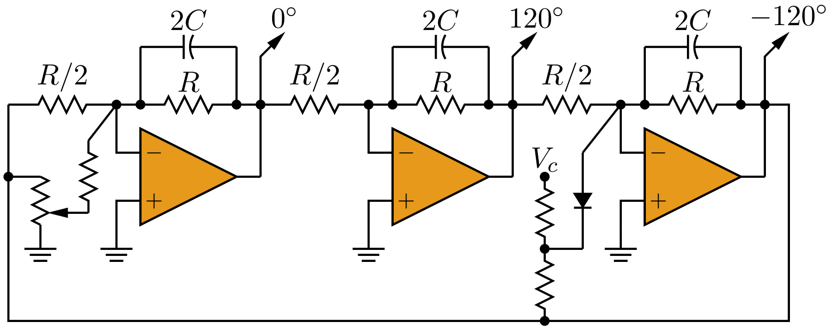


Figure 49: Three-phase oscillator [Three.m4].

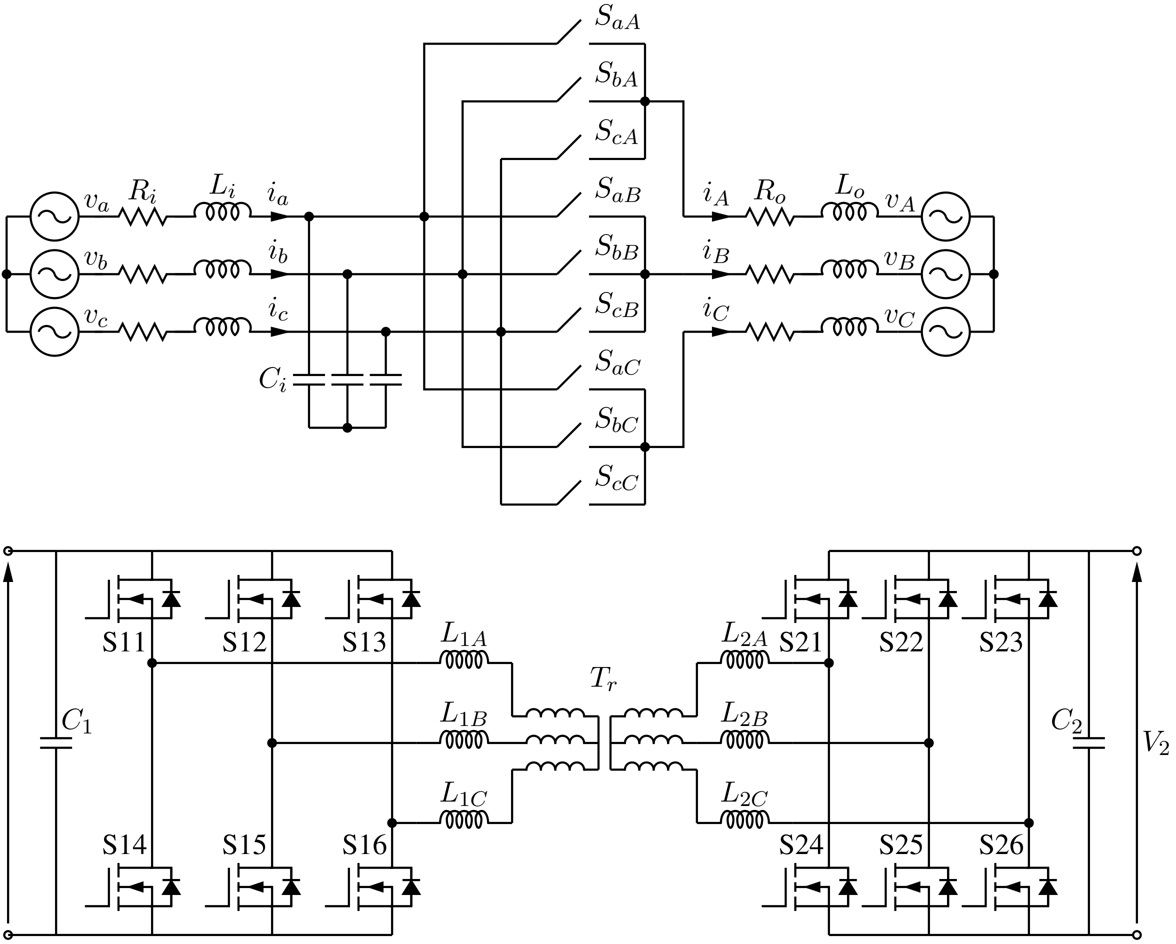


Figure 50: A three-phase switched AC-AC converter and a DC-DC converter [MC.m4].

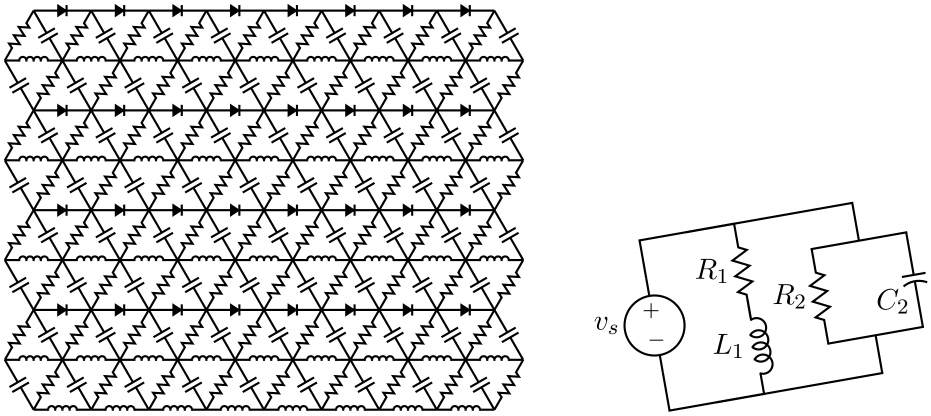


Figure 51: A repetitive network created by Pic looping and a skewed circuit used to test the macro parallel_ [ex17.m4].

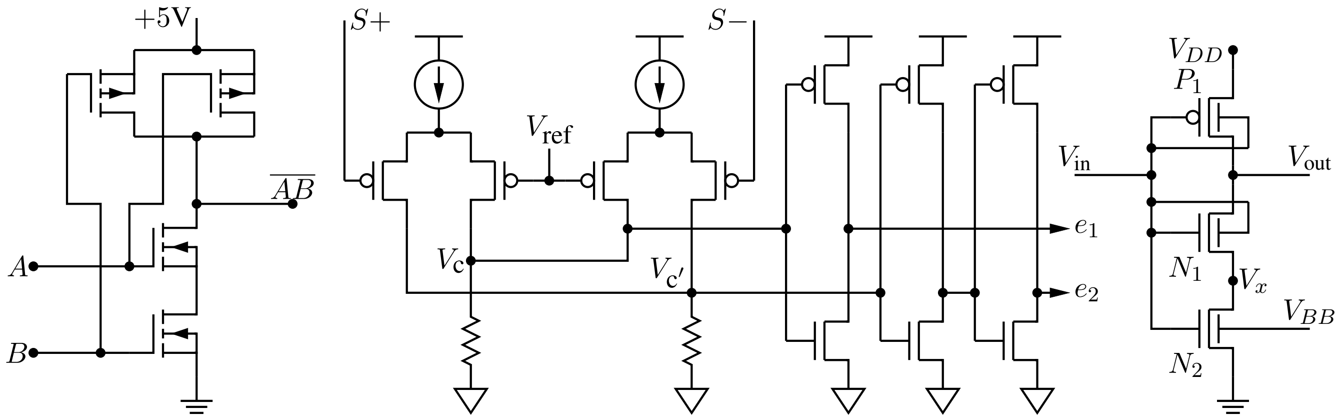


Figure 52: A CMOS NAND gate, a test circuit, and an XMOSFET example `[ex12.m4]`.

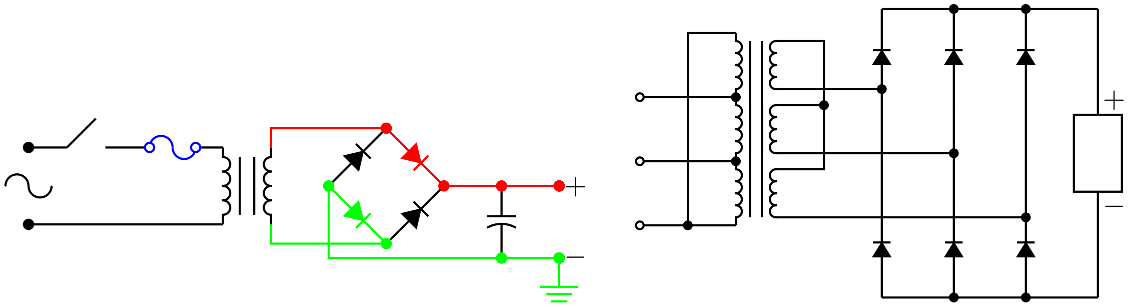


Figure 53: An elementary power supply circuit with colored elements, and a multiple-winding transformer with 3-phase rectifier `[pwrsupply.m4]`.

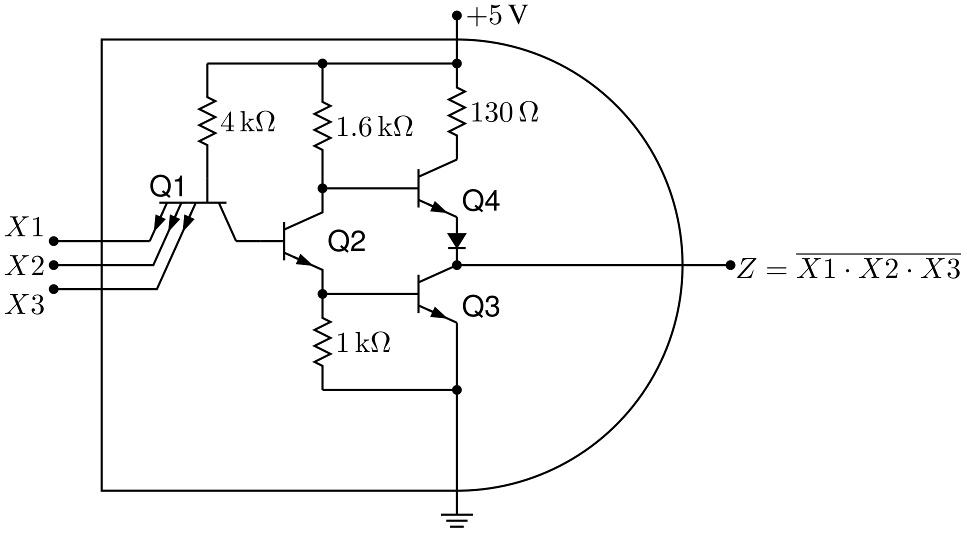


Figure 54: TTL NAND gate illustrating a transistor with multiple emitters `[TTLnand.m4]`.

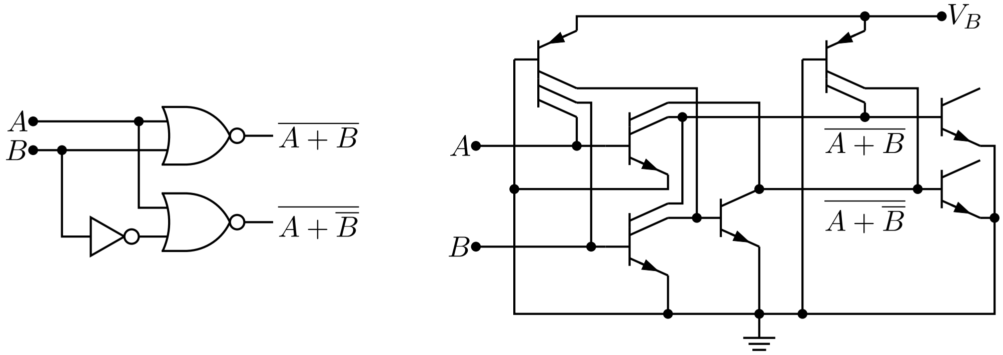


Figure 55: Gate circuit and equivalent embedded I^2L components illustrating multiple collectors `[I2L.m4]`.

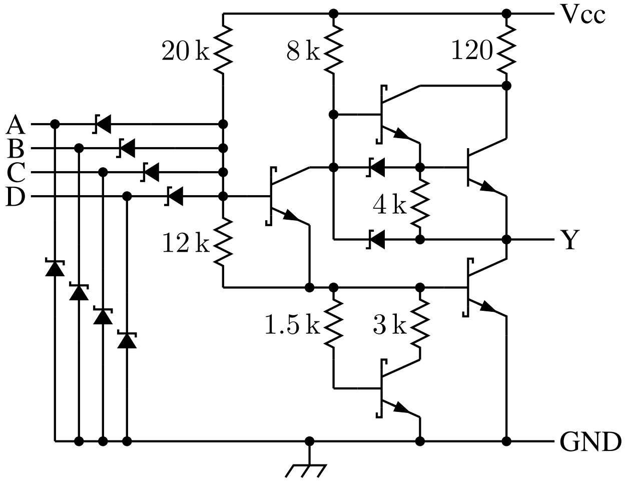


Figure 56: A 4-input NAND circuit illustrating the `S` (Schottky) option of `bi_trans` `[Schottky.m4]`.

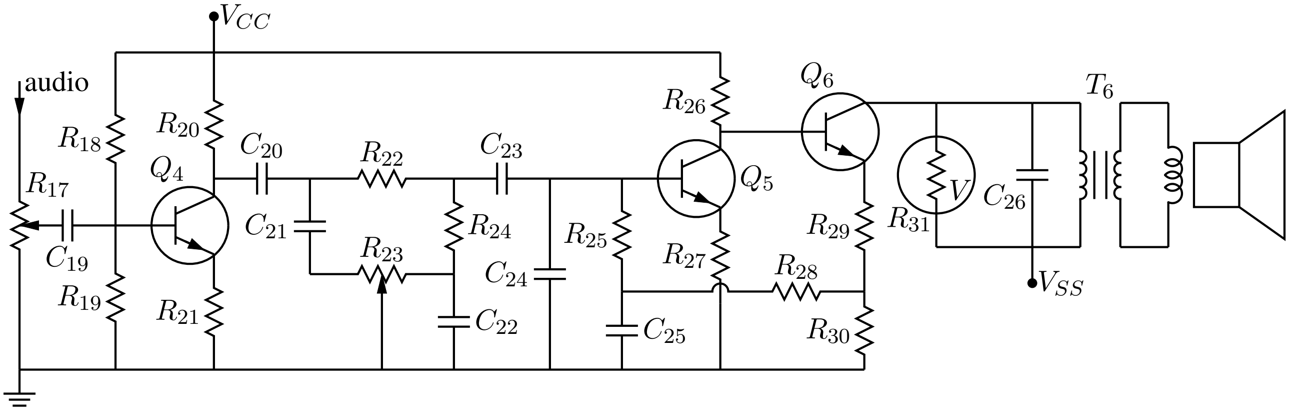


Figure 57: Transistor radio audio chain `[ex11.m4]`.

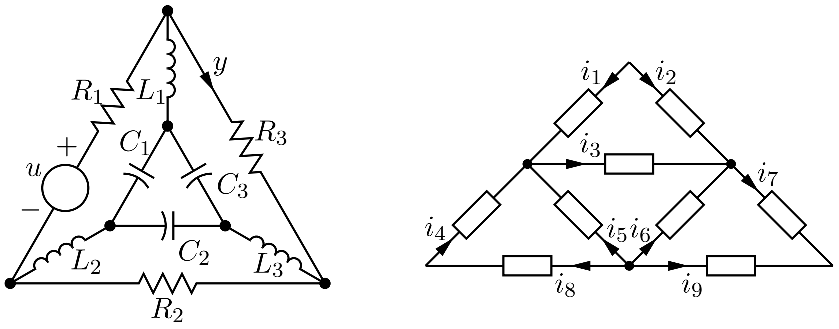


Figure 58: Labels on non-manhattan elements [ex04.m4].

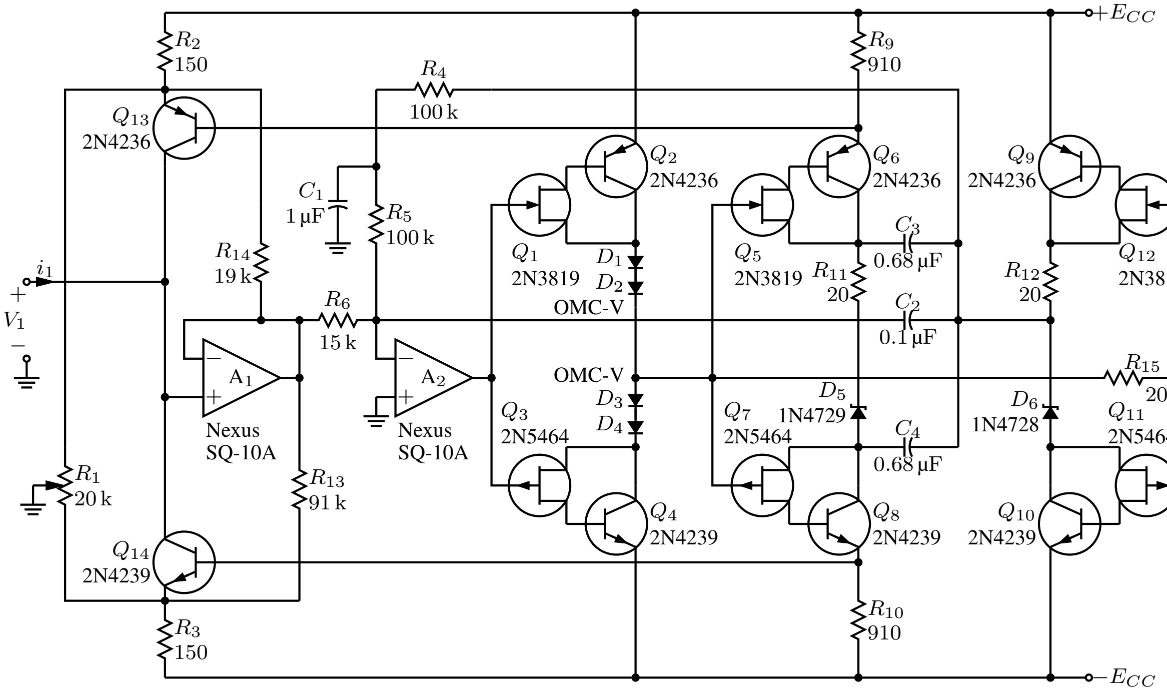


Figure 59: Realization of a controlled source (illustrating stacked element labels) [Csource.m4].

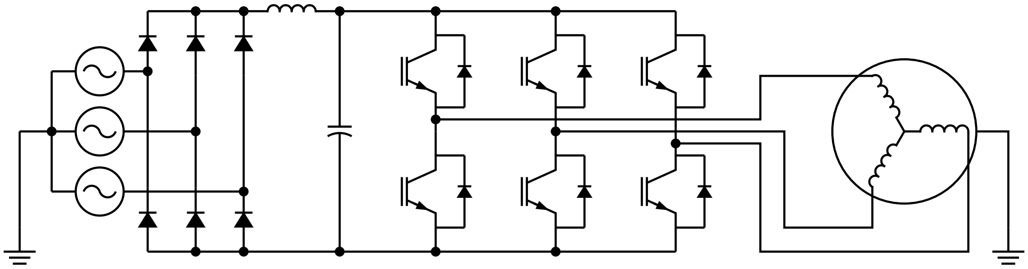


Figure 60: Synchronous machine driven by variable-speed drive and rectifier [Drive.m4].

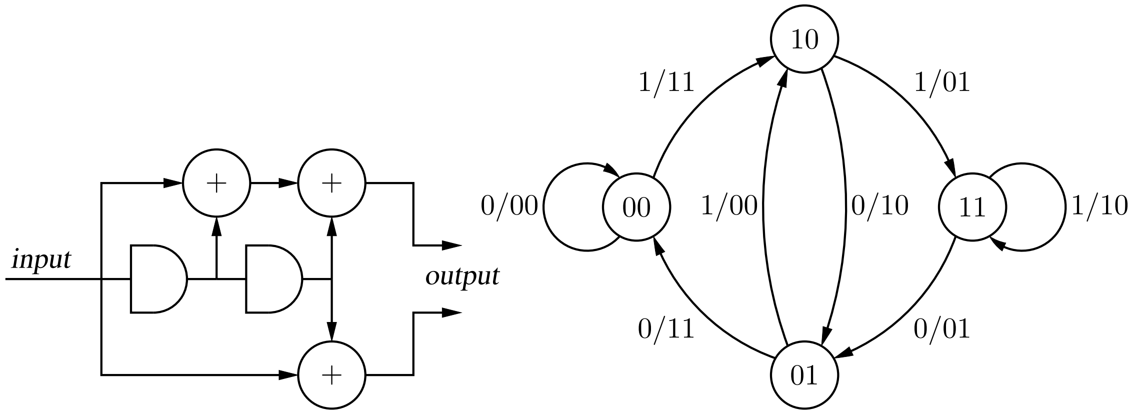


Figure 61: A rate $1/2$ binary convolutional coder and its state diagram [ex16.m4].

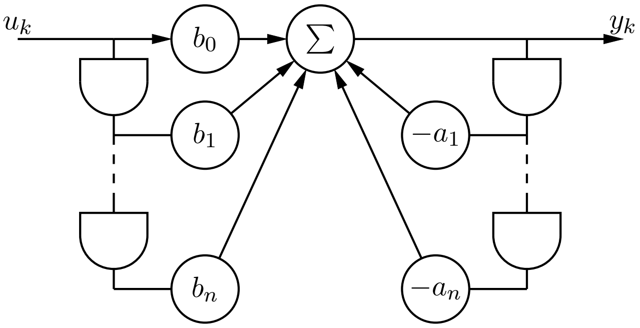


Figure 62: Digital filter [ex03.m4].

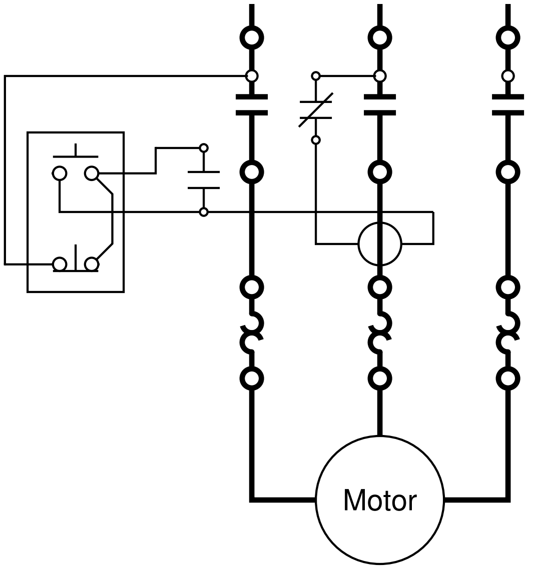


Figure 63: Motor control connections [MotorControl.m4].

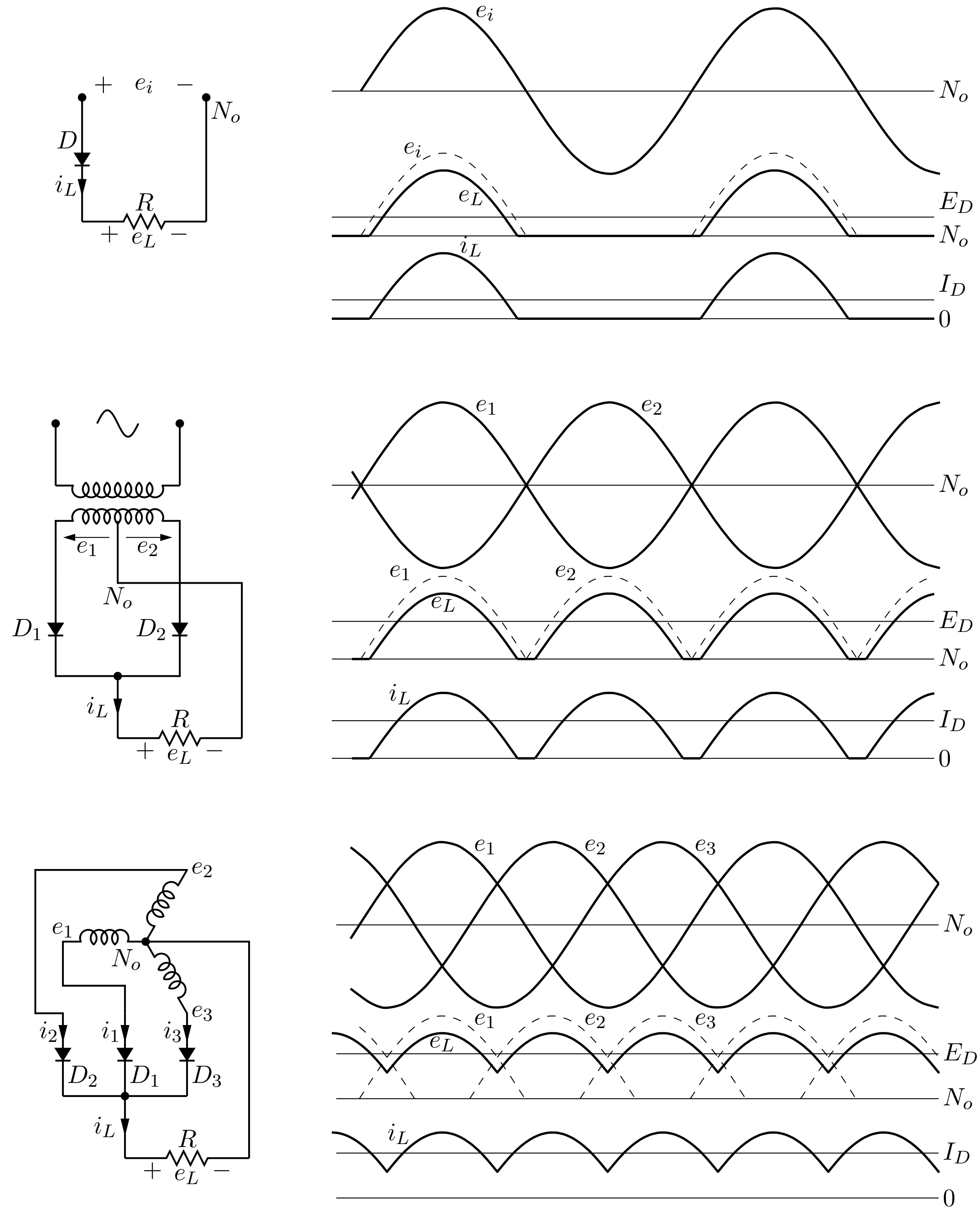


Figure 64: Rectifier circuits and waveforms `[Rectifiers.m4]`.

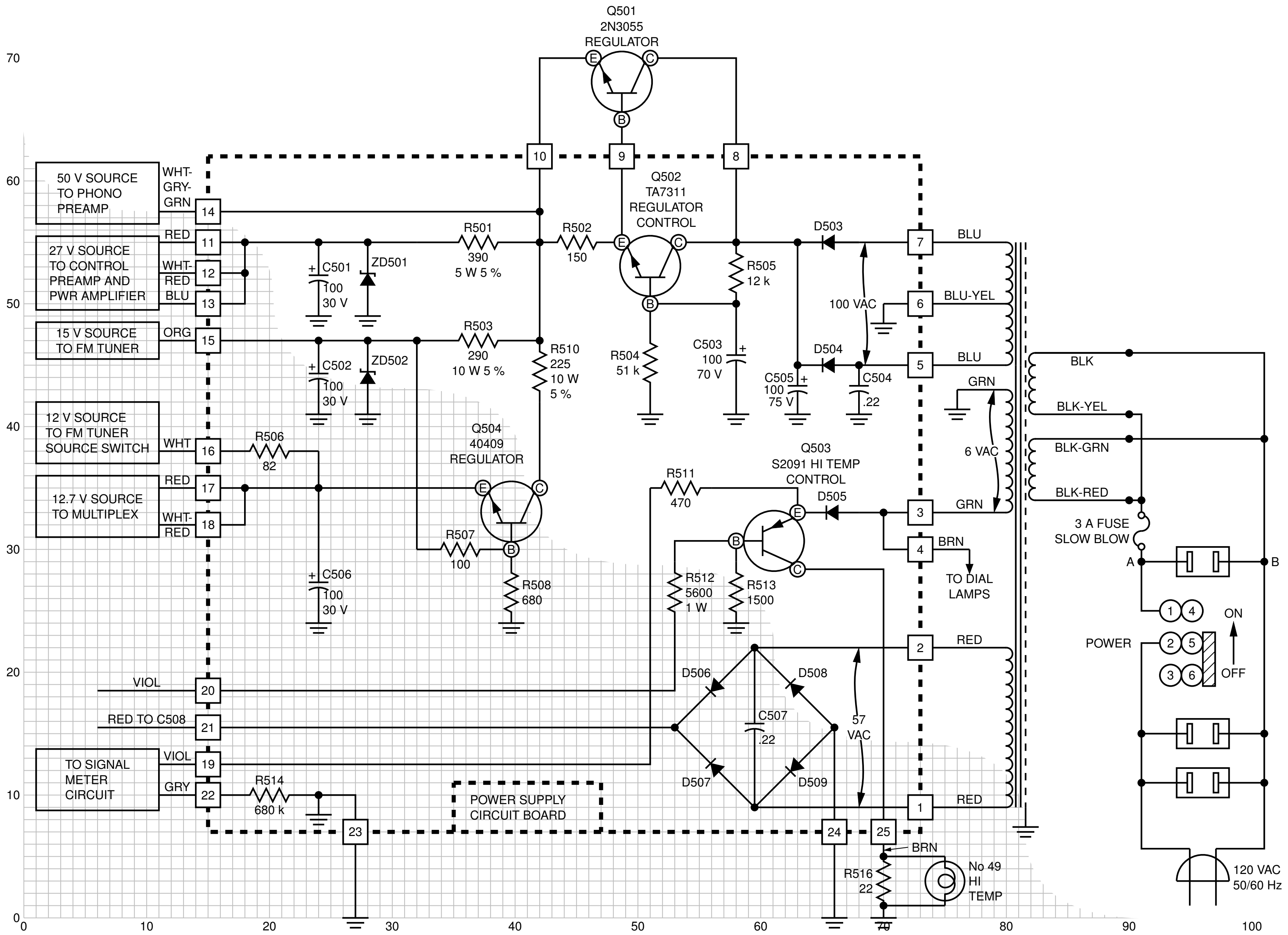


Figure 65: The power supply of a Heathkit AR-15 (Now, *that* was a receiver!) with custom transformer and other elements, drawn on a grid (partially shown) to aid in placement [Heathkit.m4].

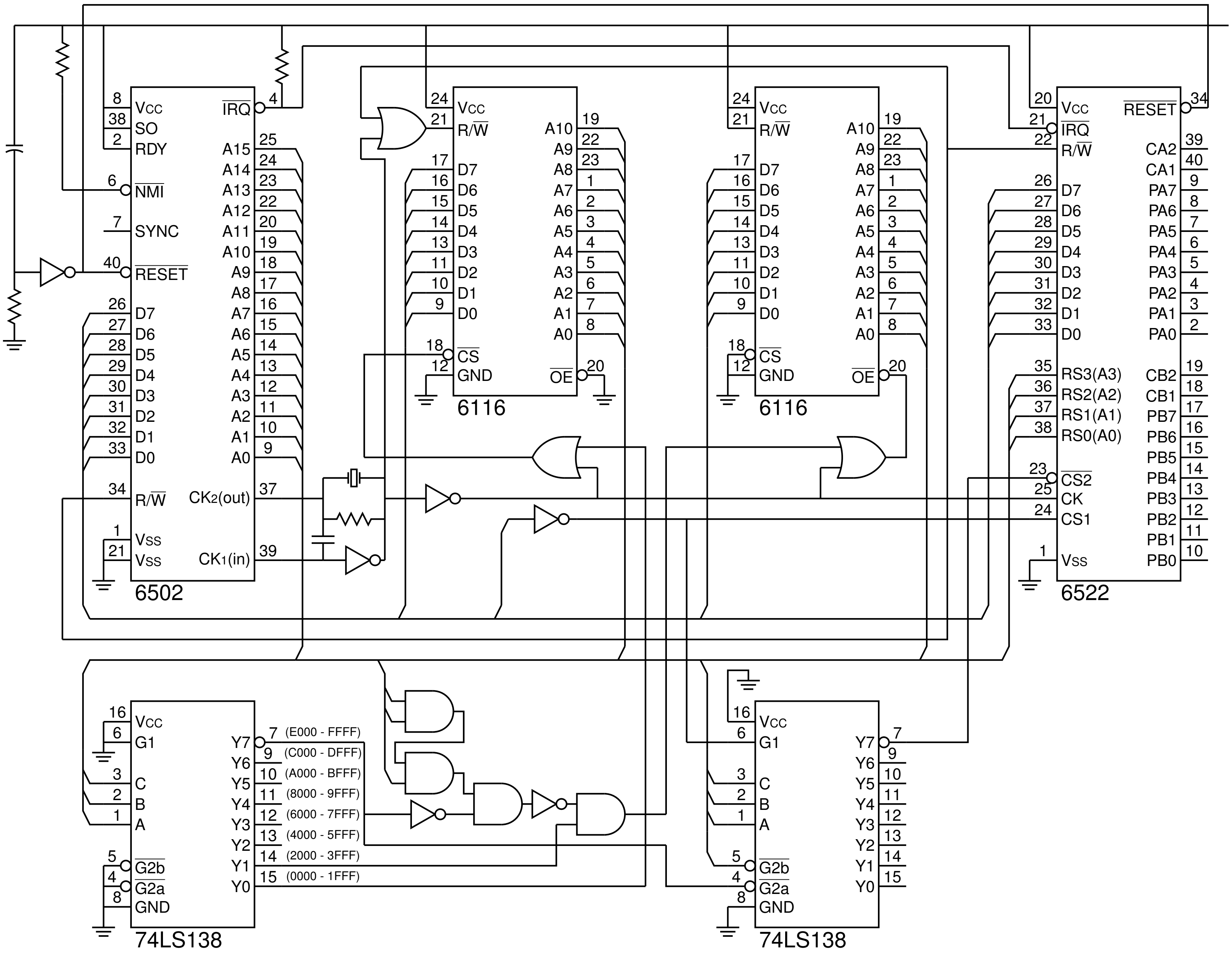


Figure 66: A digital circuit of moderate size, redrawn from M. P. Maclenan and G. M. Burns, “An Approach to Drawing Circuit Diagrams for Text Books,” Tugboat (12)1, March 1991, pp. 66-69 `[lcct.m4]`.

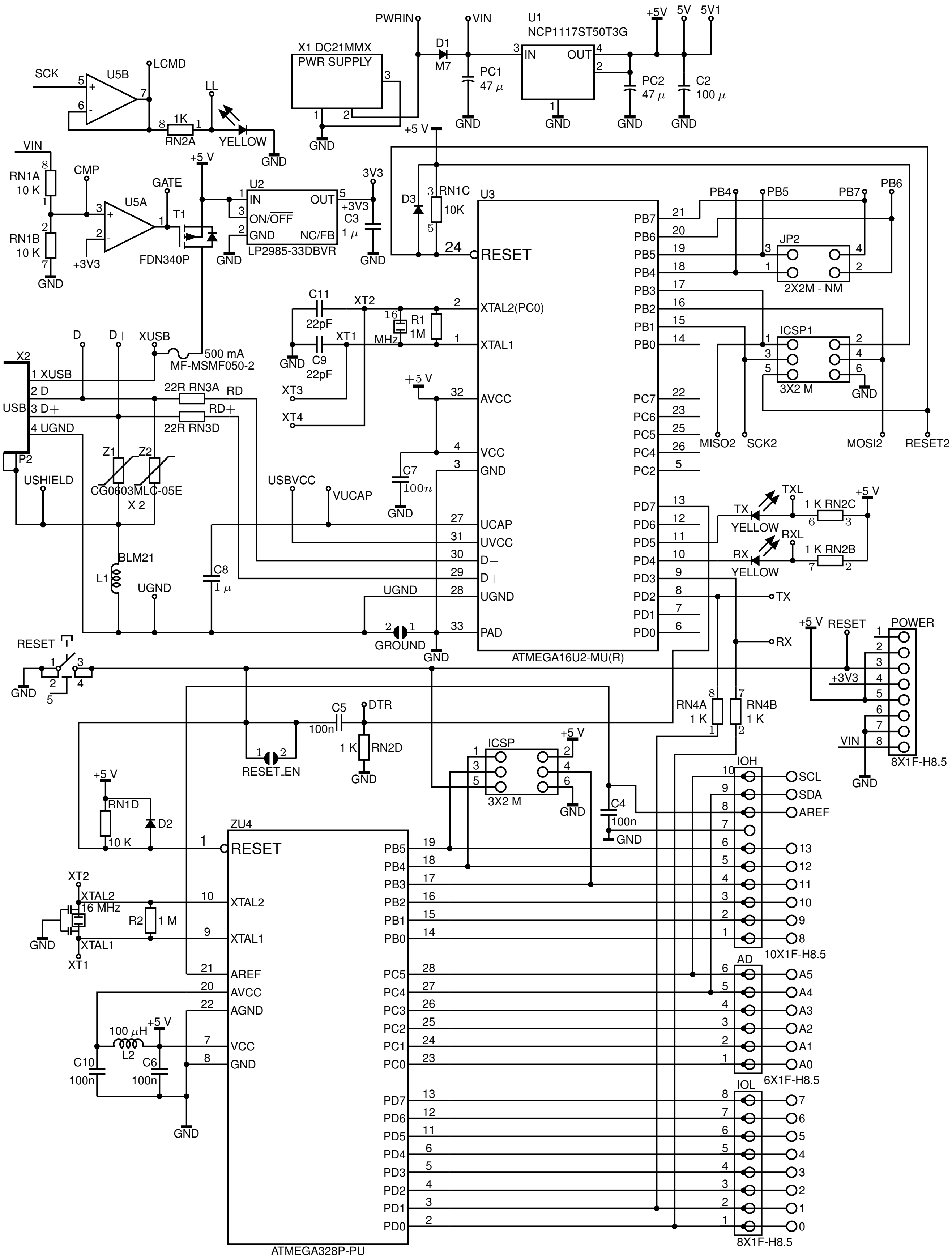


Figure 67: An Arduino UNO circuit adapted and redrawn [UNO.m4].

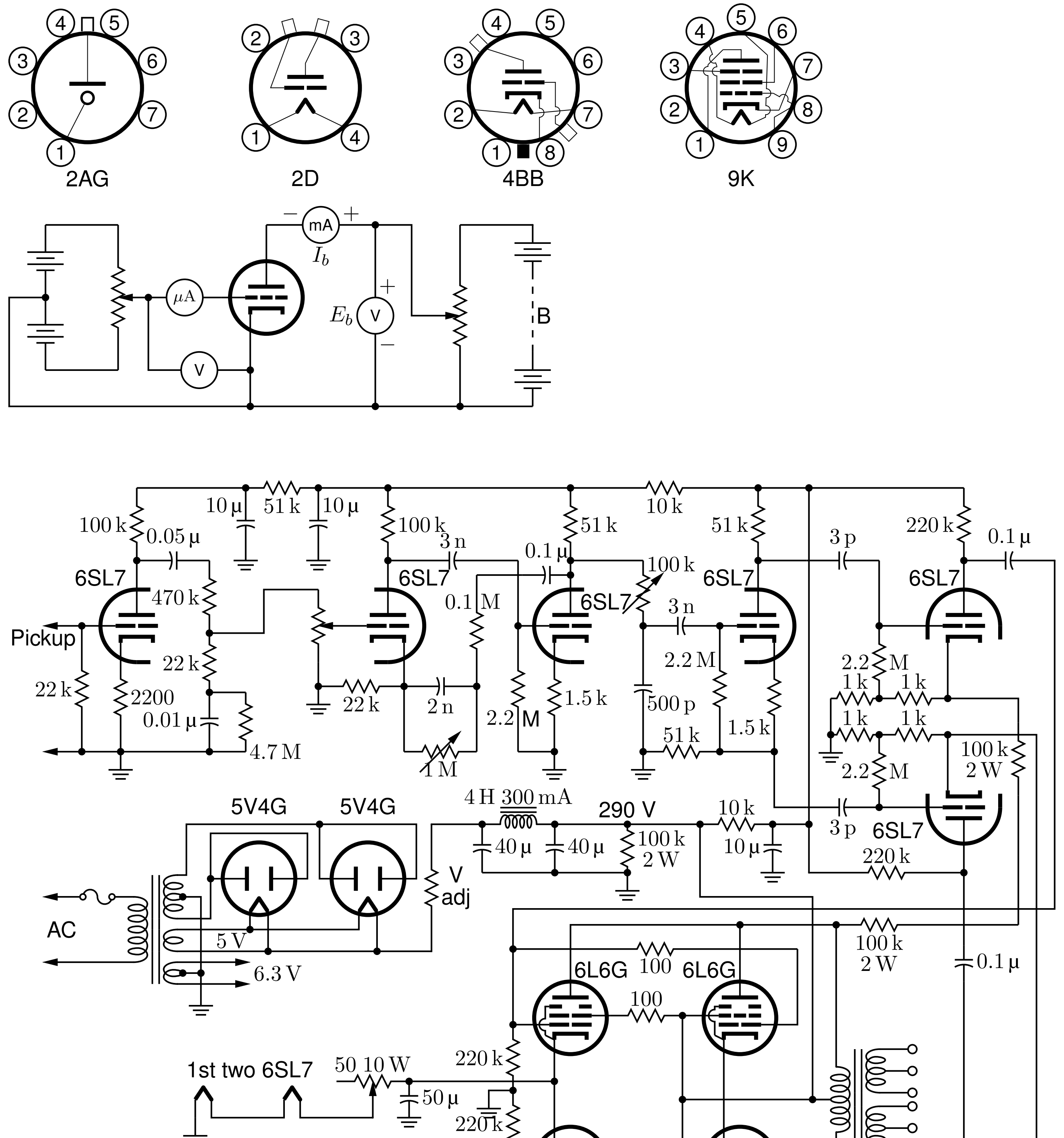


Figure 68: Electron-tube diagrams: a few bottom-view base diagrams, a generic triode test circuit, and a 25-watt audio amplifier adapted from F. Langford-Smith, *Radiotron Designer's Handbook,* fourth edition, Harrison, NJ: Radio Corporation of America, 1952 [Tubediags.m4].

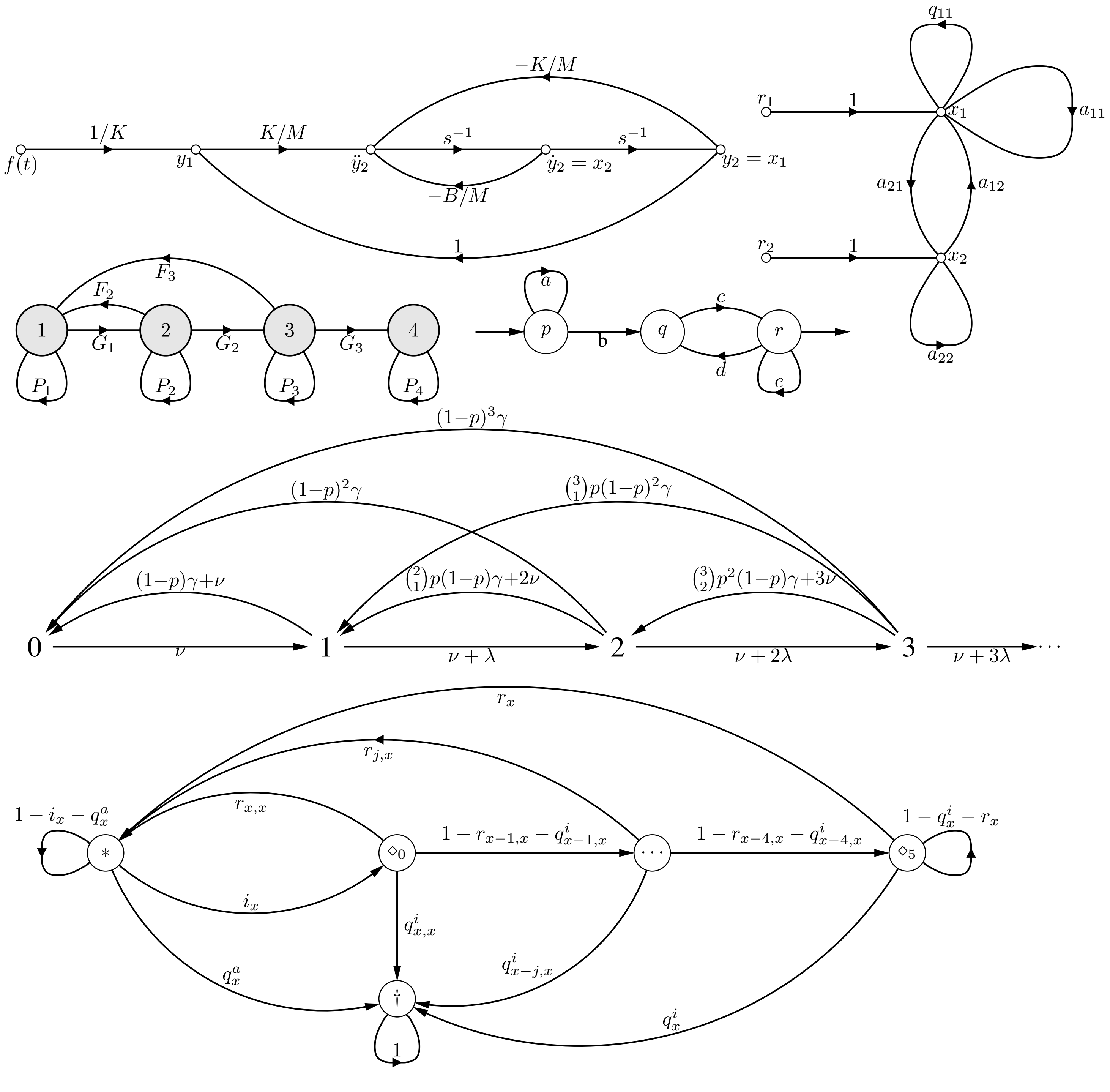


Figure 69: Signal-flow graphs [sfg.m4].

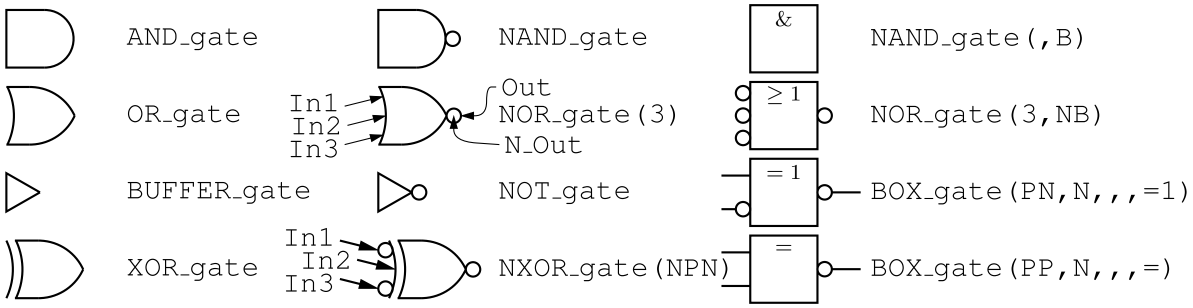


Figure 70: Basic logic gates [Logic.m4].

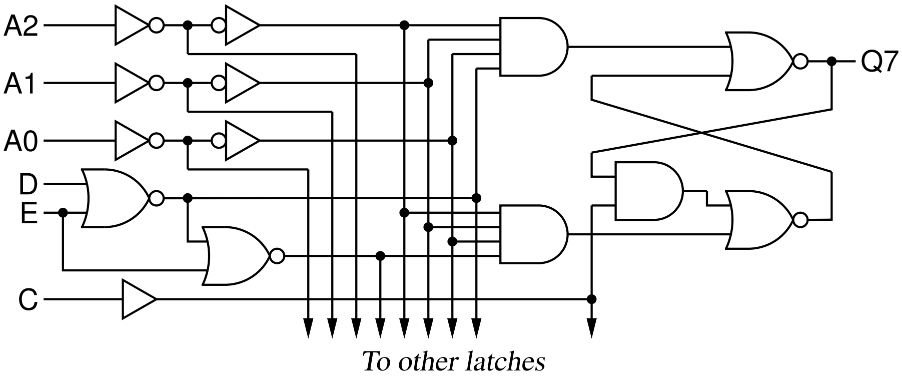


Figure 71: General-purpose latch: a small logic circuit [ex08.m4].

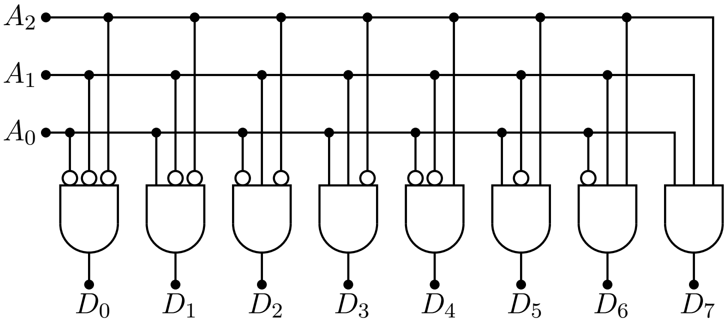


Figure 72: Decoder logic, constructed using the for_ macro [Decoder.m4].

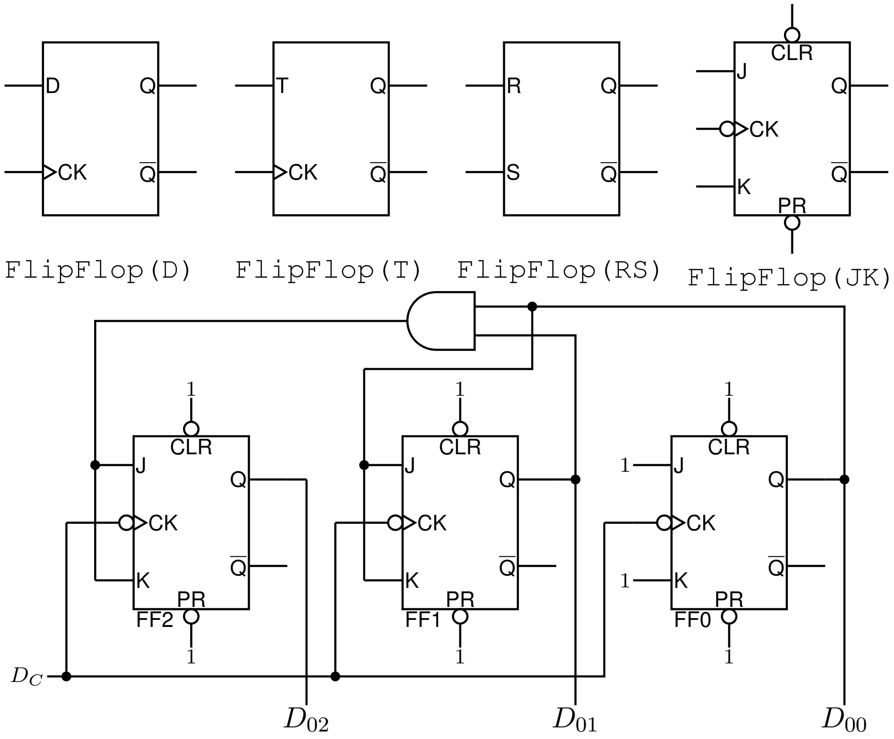


Figure 73: Some flip-flops [ex21.m4].

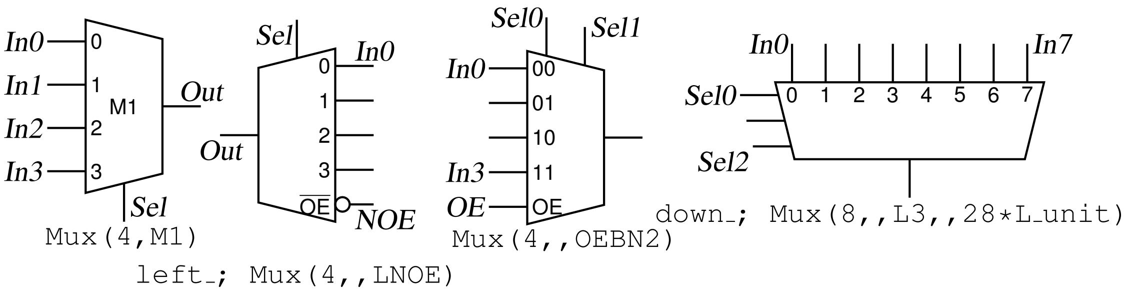


Figure 74: Multiplexer [Multiplexer.m4].

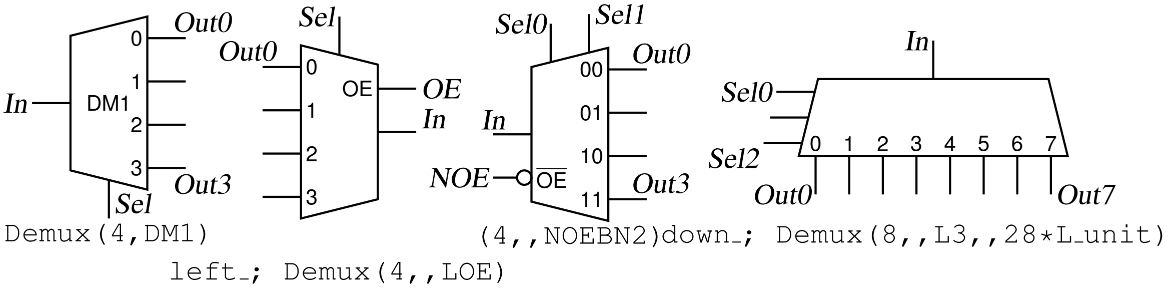


Figure 75: Demultiplexer [Demultiplexer.m4].

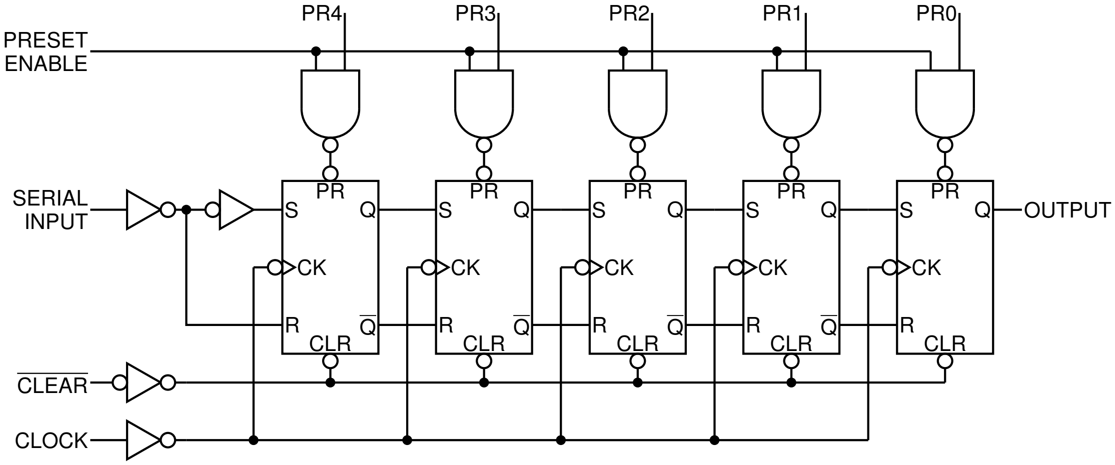
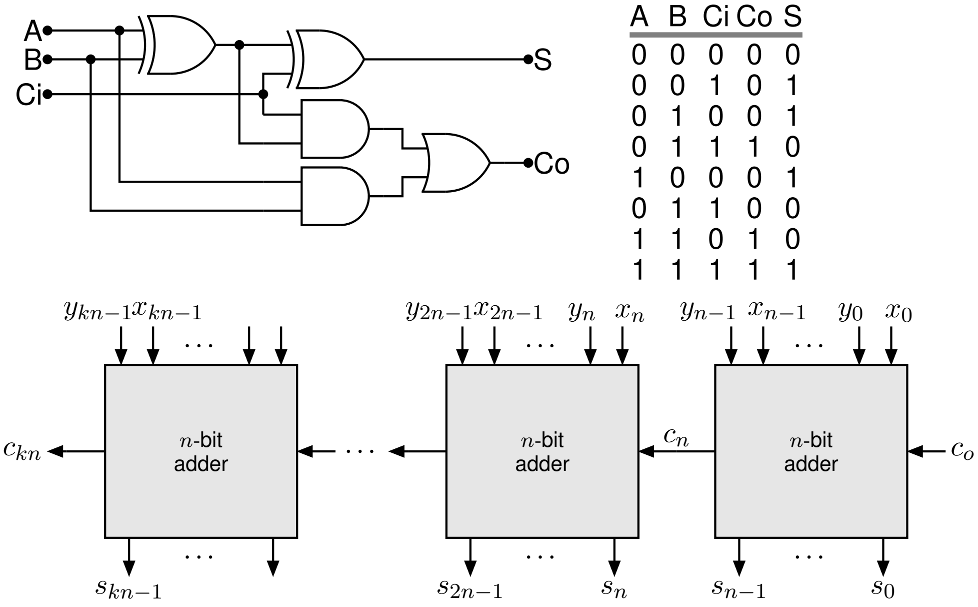
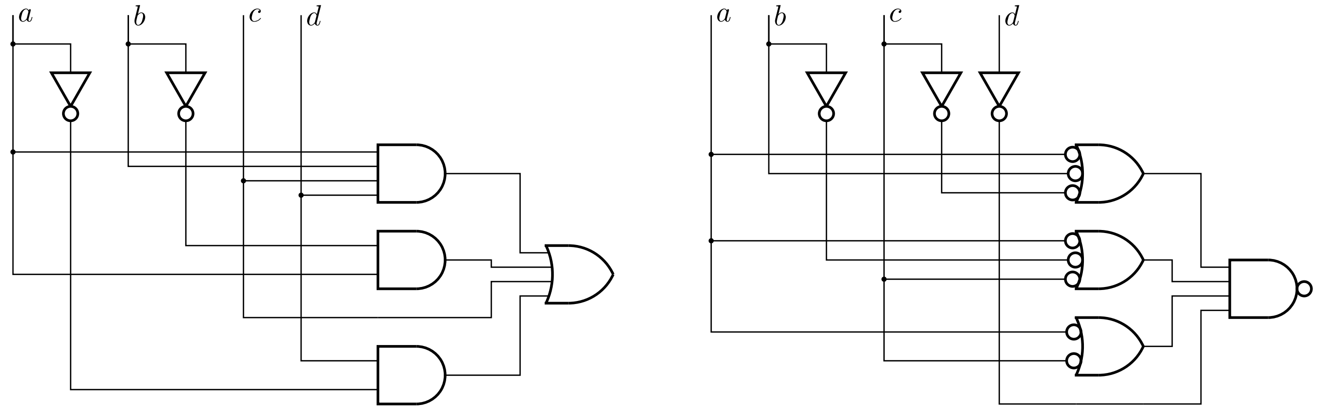


Figure 76: A 5-bit shift register drawn using a custom flip-flop [ShiftR.m4].



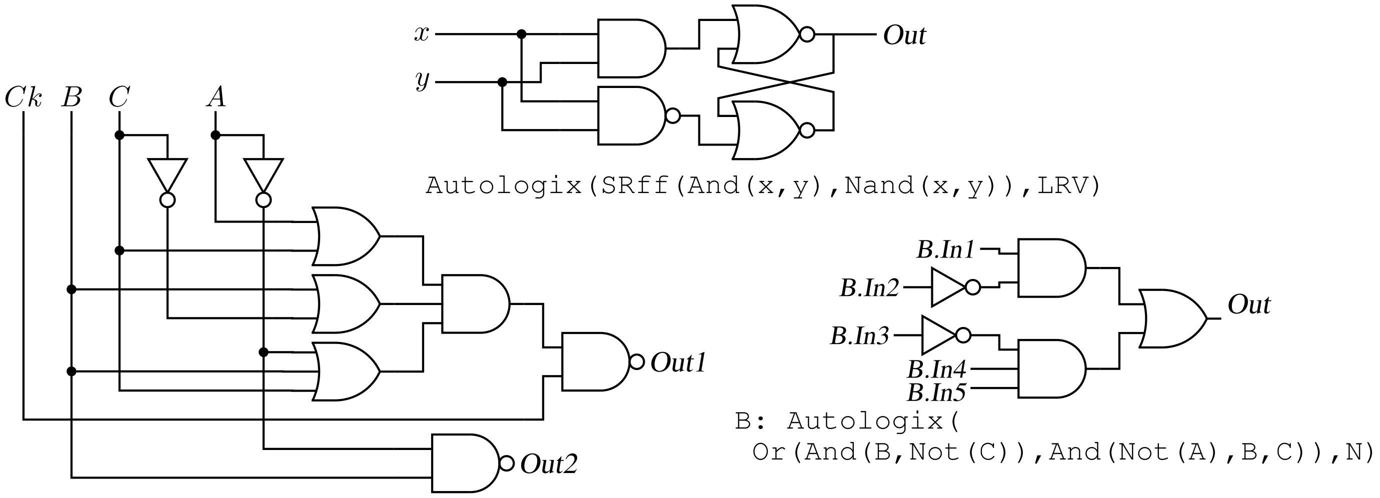
A	B	Ci	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
0	1	1	0	0
1	1	0	1	0
1	1	1	1	1

Figure 77: A full adder and a cascade of n-bit adders [Adder.m4].



```
CanLogic(AND,,OR,,abcd,~ba,c,d~a)        CanLogic(OR,N,NAND,,ab~c,a~bc,ac,~d)
```

Figure 78: A way of automatically drawing two-layer logic diagrams [CanLogic.m4].

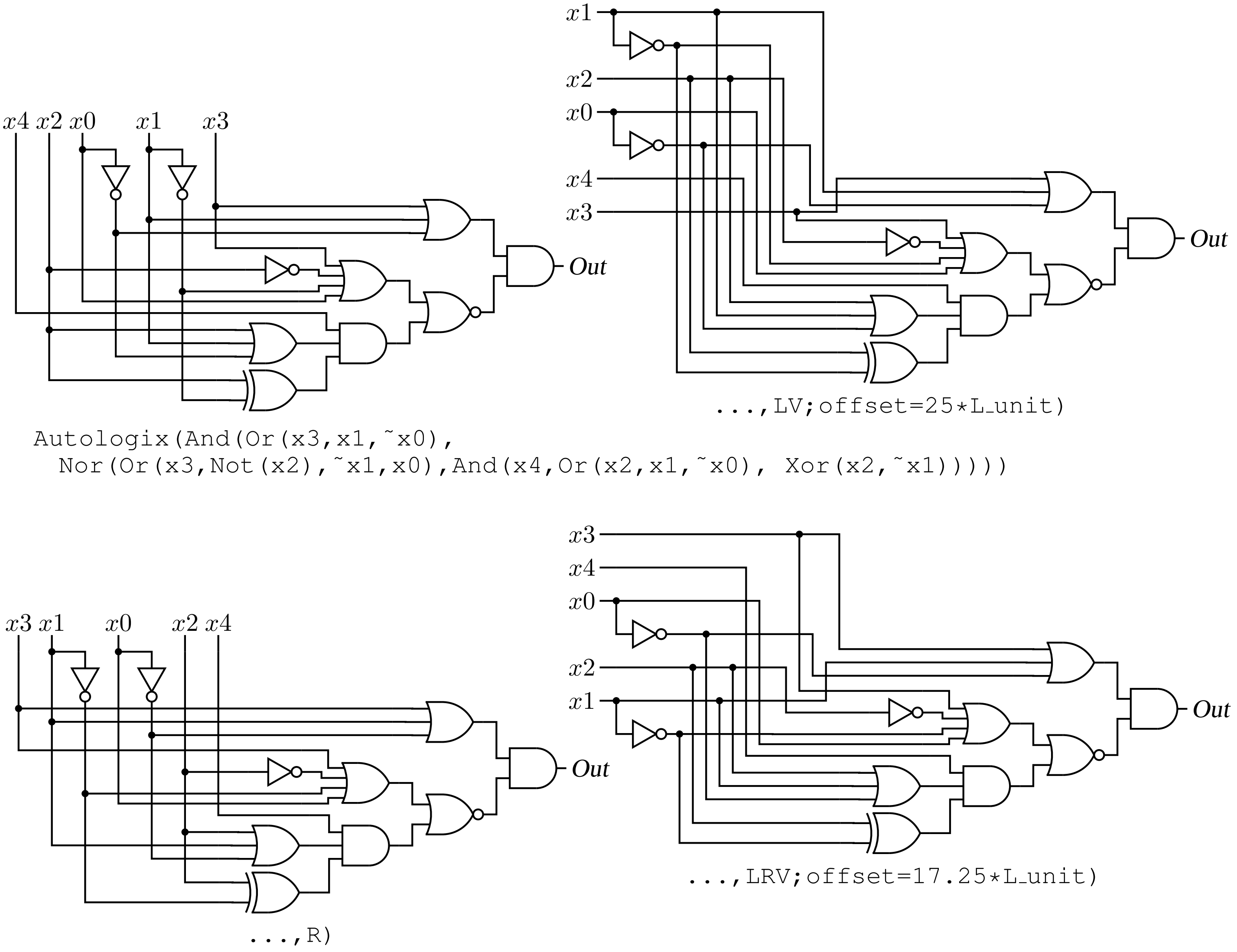


```
Autologix(SRff(And(x,y),Nand(x,y)),LRV)
```

```
B: Autologix(
  Or(And(B,Not(C)),And(Not(A),B,C)),N)
```

```
Autologix(
  Nand(And(Or(A,C),Or(B,~C),Or(~A,B,C)),Ck);
  Nand(~A,B))@with .n at last [].s+(0,-2bp__))
```

Figure 79: 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].



```
...,LV;offset=25*L_unit)
```

```
Autologix(And(Or(x3,x1,~x0),
  Nor(Or(x3,Not(x2),~x1,x0),And(x4,Or(x2,x1,~x0), Xor(x2,~x1)))))
```

```
...,LRV;offset=17.25*L_unit)
```

```
...,R)
```

Figure 80: 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]`.

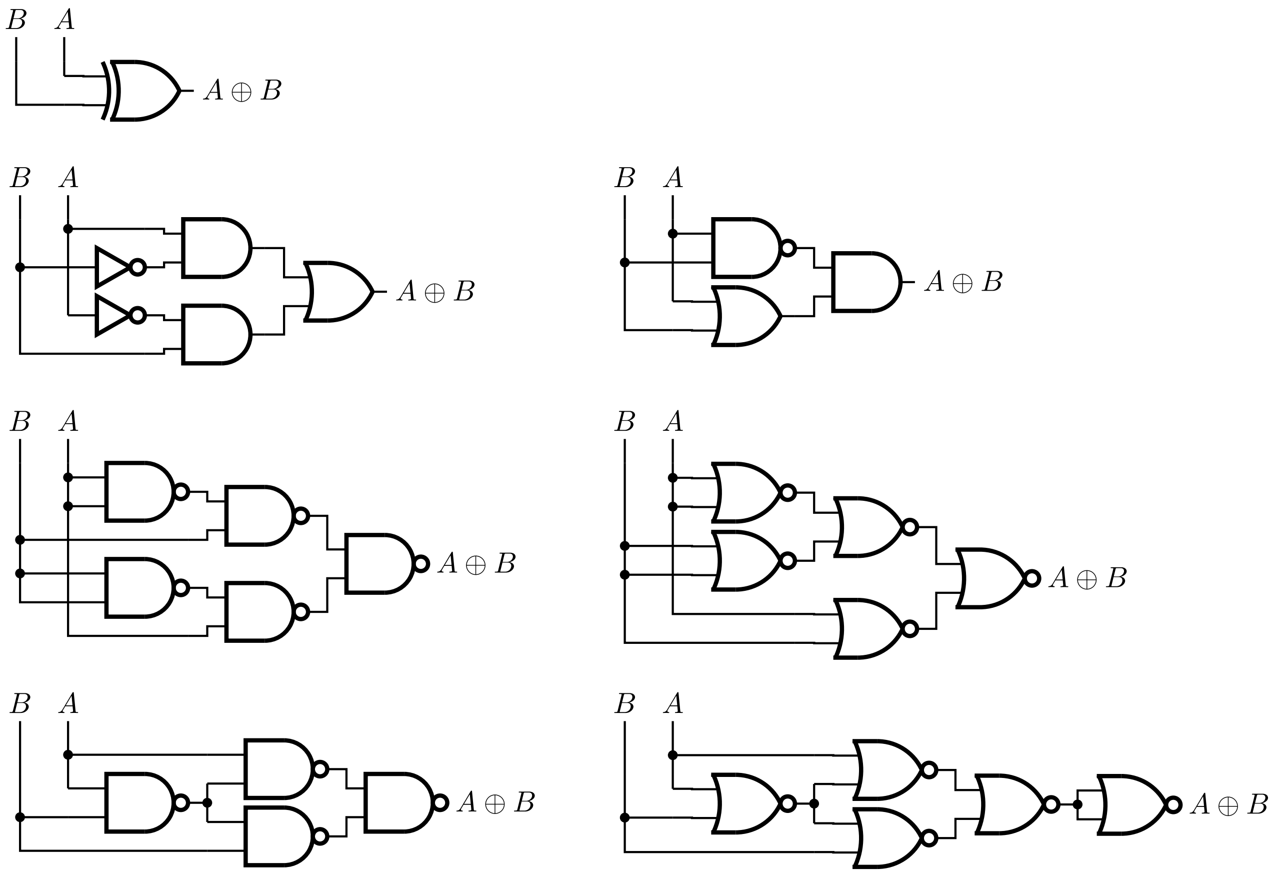


Figure 81: Realizations of the XOR function using `Autologix` `[XOR.m4]`.

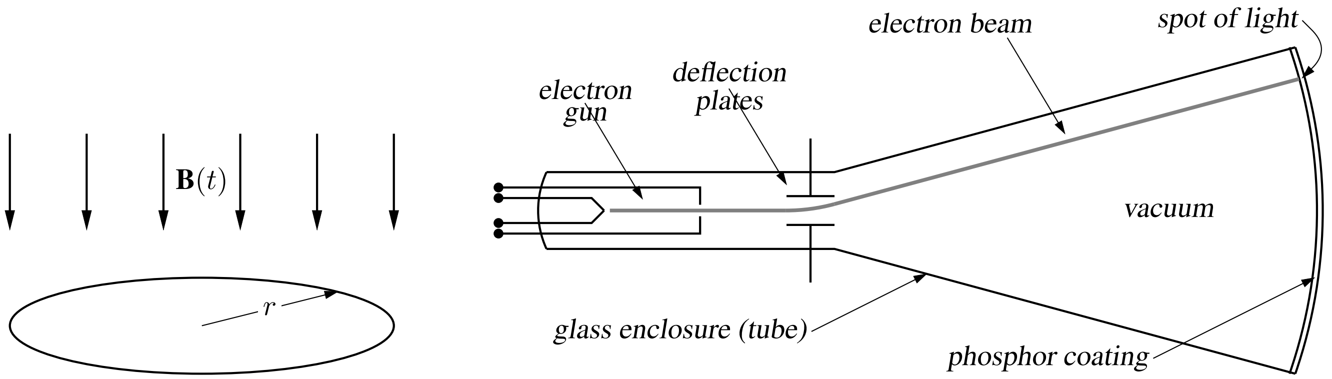


Figure 82: Line diagrams `[ex00.m4]`.

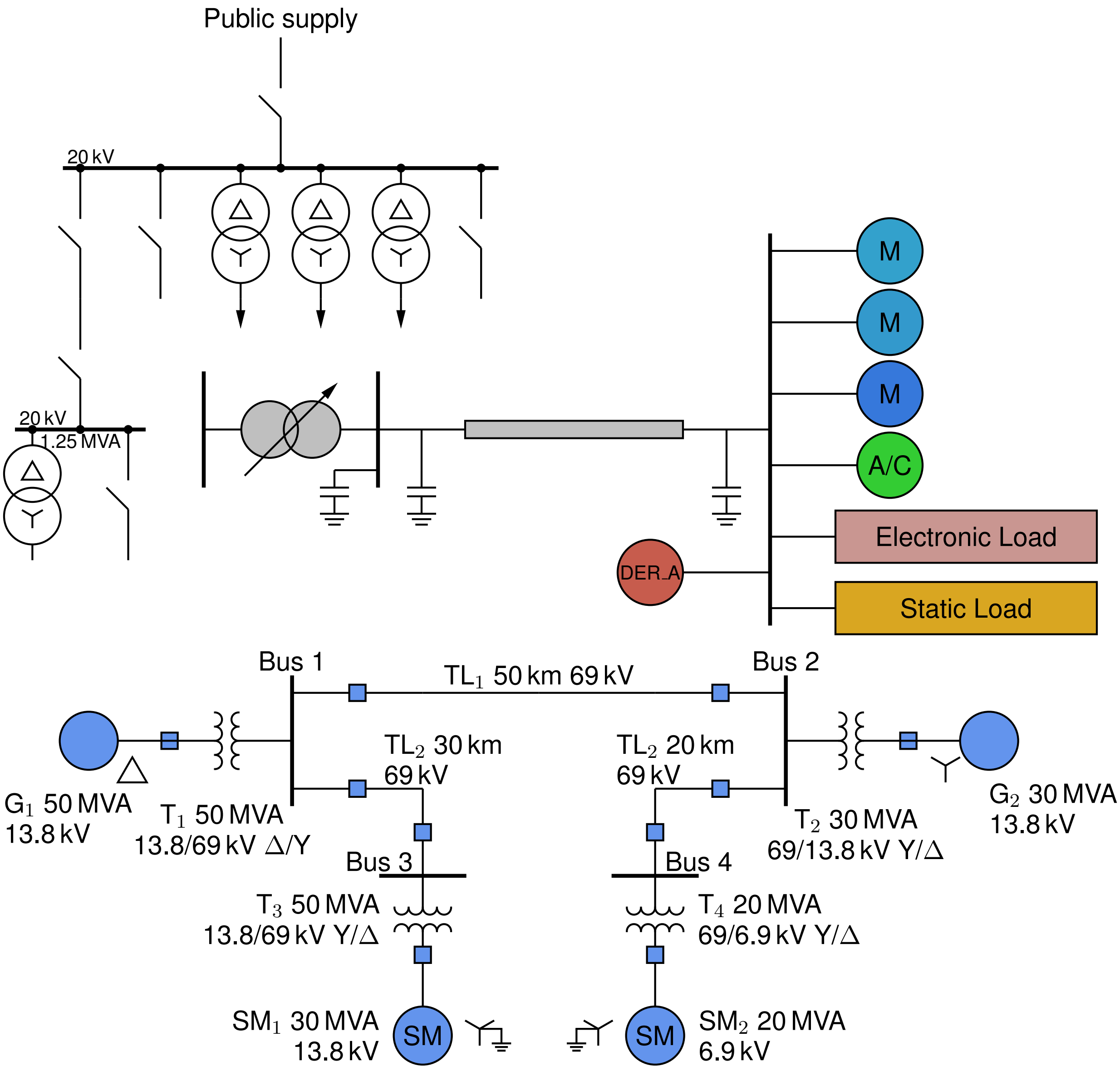


Figure 83: A test of experimental single-line diagram macros `[EEP.m4]`.

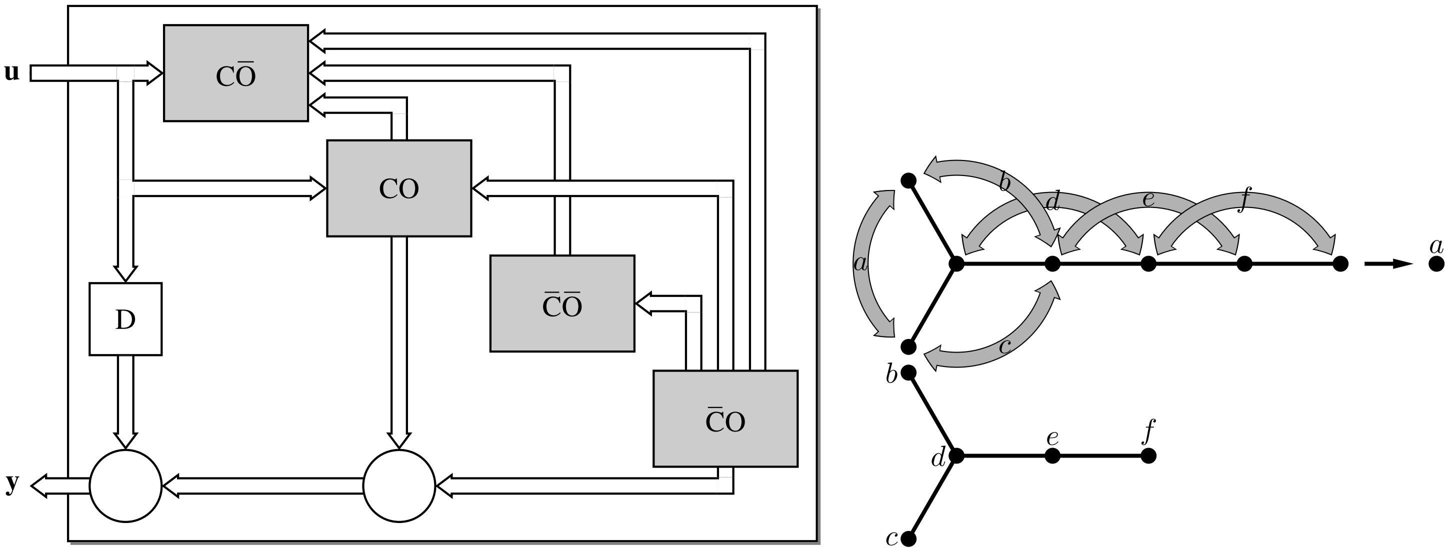


Figure 84: Use of `darrow` and `Darc` `[ex05.m4]`.

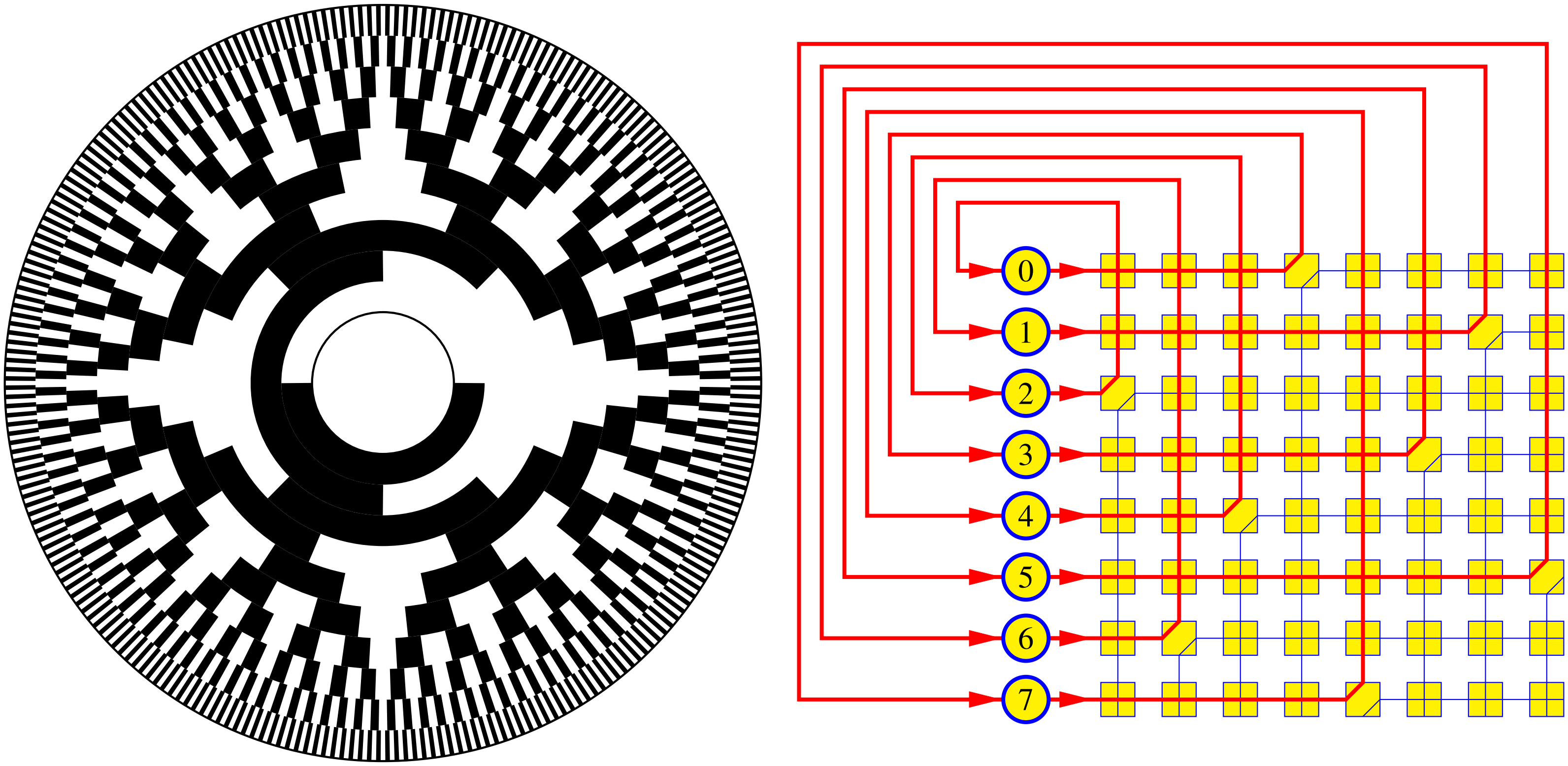
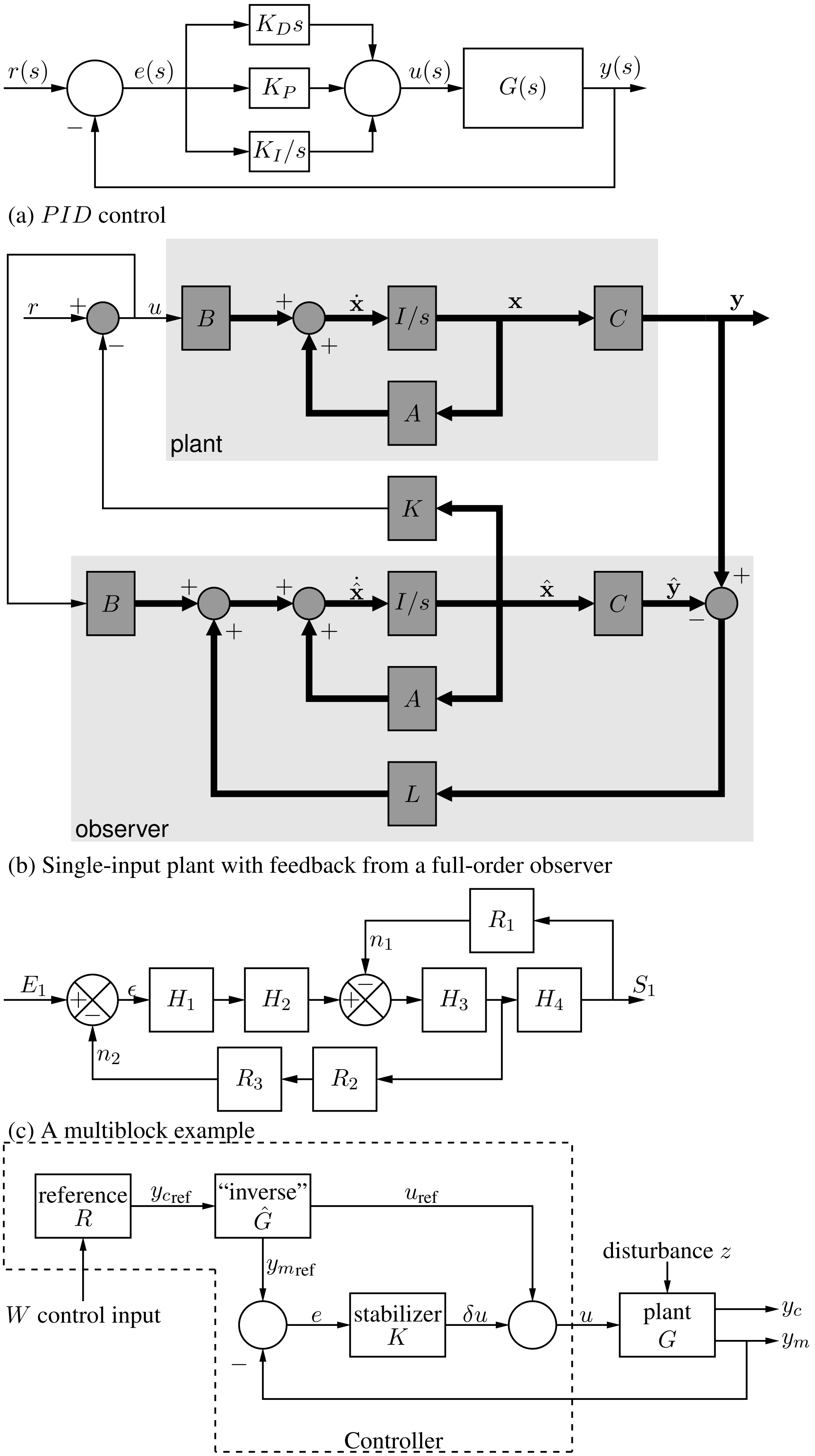


Figure 85: Gray code 10-bit encoder disk pattern and a crossbar switch `[GrayCode.m4]`.



(a) PID control

(b) Single-input plant with feedback from a full-order observer

(c) A multiblock example

(d) Nonlinear feedforward (for performance) and small-signal feedback (for stability)

Figure 86: Control-system block diagrams [control.m4]

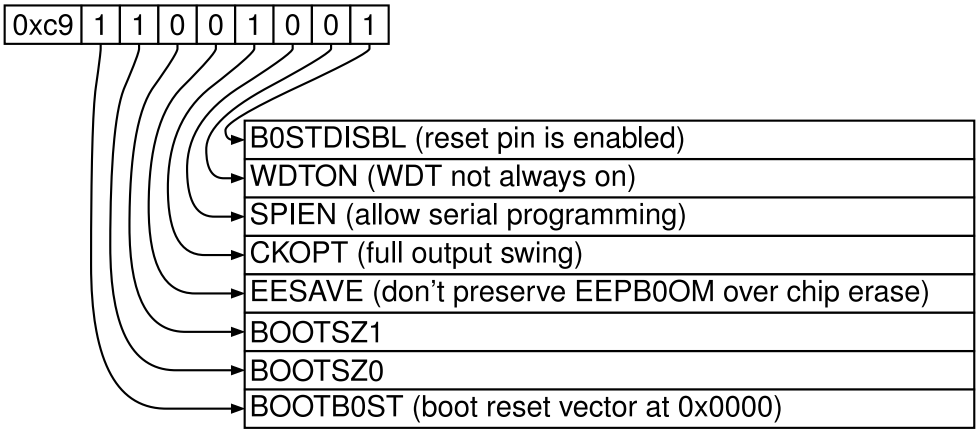

0xc9
1
1
0
0
1
0
0
1
B0STDISBL (reset pin is enabled)
WDTON (WDT not always on)
SPIEN (allow serial programming)
CKOPT (full output swing)
EESAVE (don't preserve EEPB0OM over chip erase)
BOOTSZ1
BOOTSZ0
BOOTB0ST (boot reset vector at 0x0000)

Figure 87: Elementary splines `[Byte.m4]`.

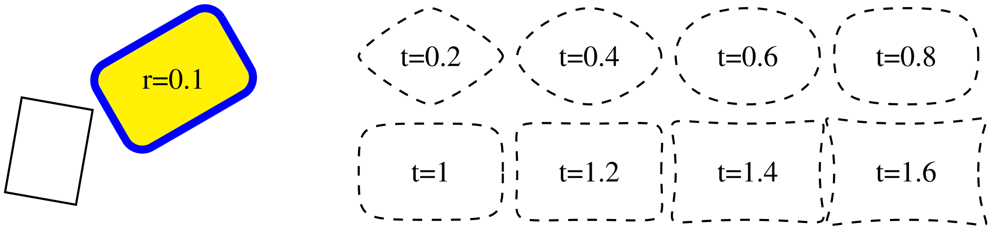


Figure 88: The macro `rotbox(`*wid,ht,type,*`[r|t=`*val*`])` draws a box in the current direction `[Rotbox.m4]`.

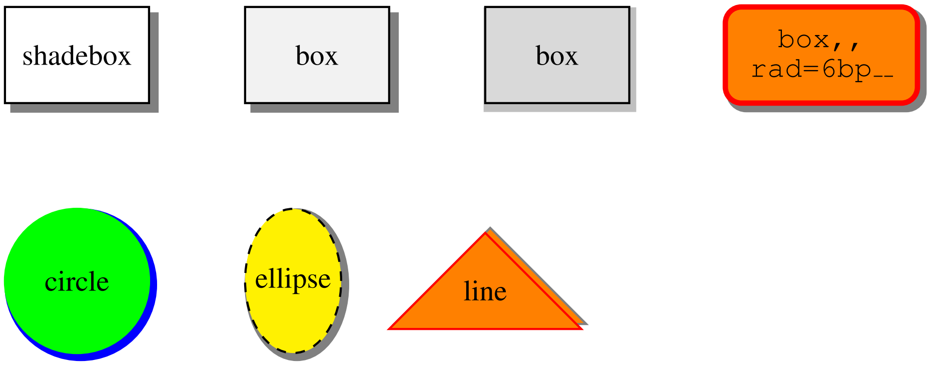


Figure 89: The macro `shadowed(box|circle|ellipse|line,[at` *position*`],`*keys*`)` shadowed object `[shadowed.m4]`.

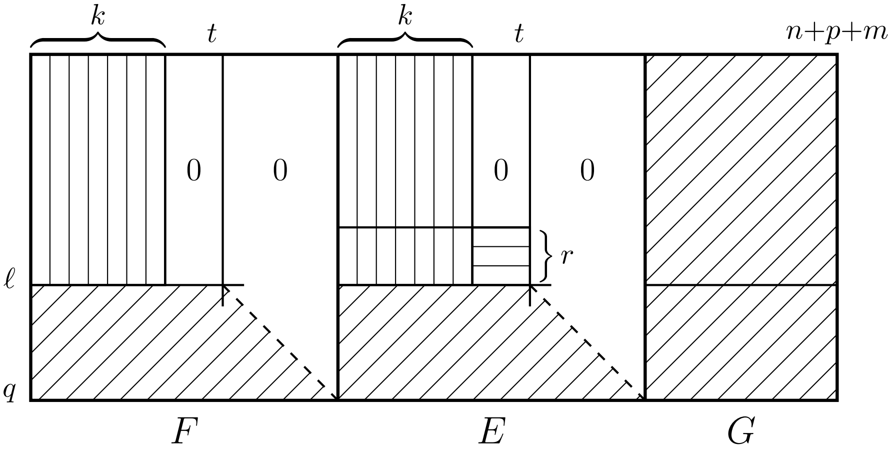


Figure 90: Crosshatching `[ex06.m4]`.

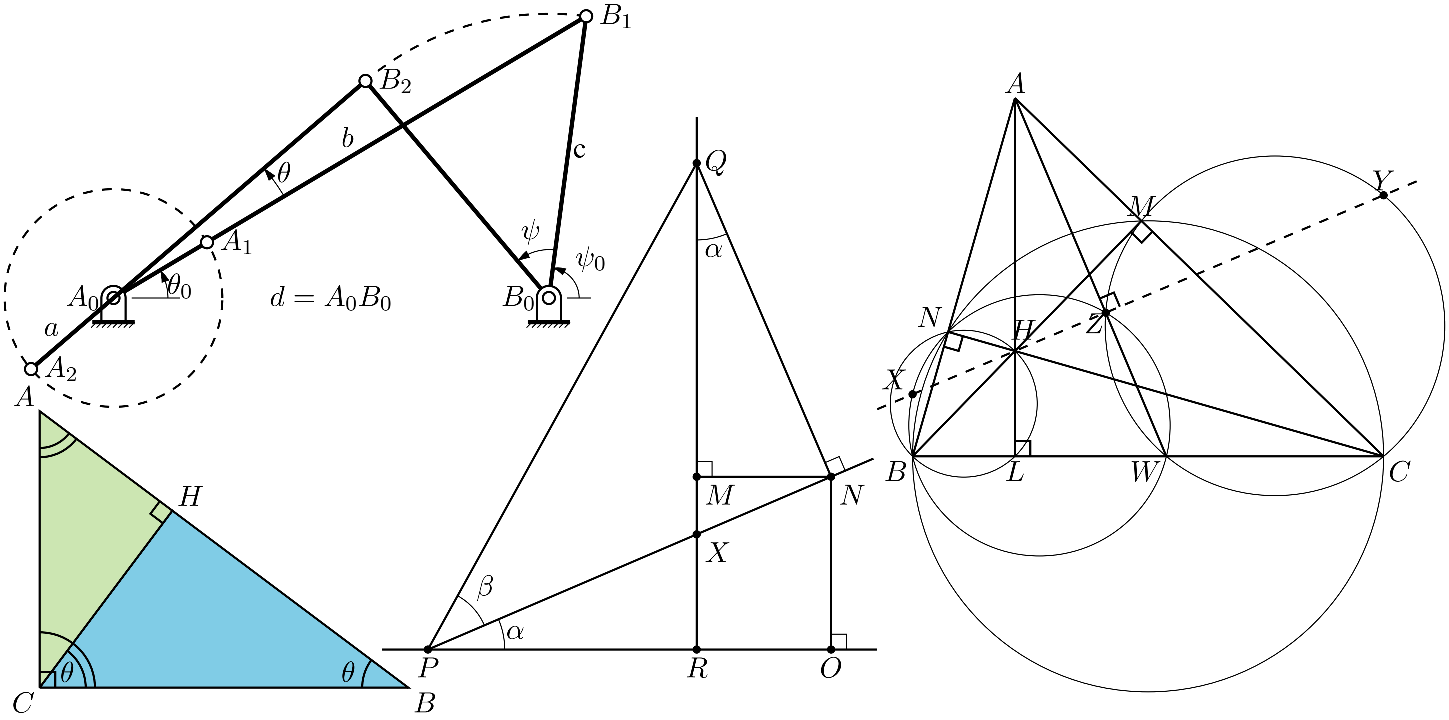


Figure 91: Some geometrical constructions `[Geometry.m4]`.

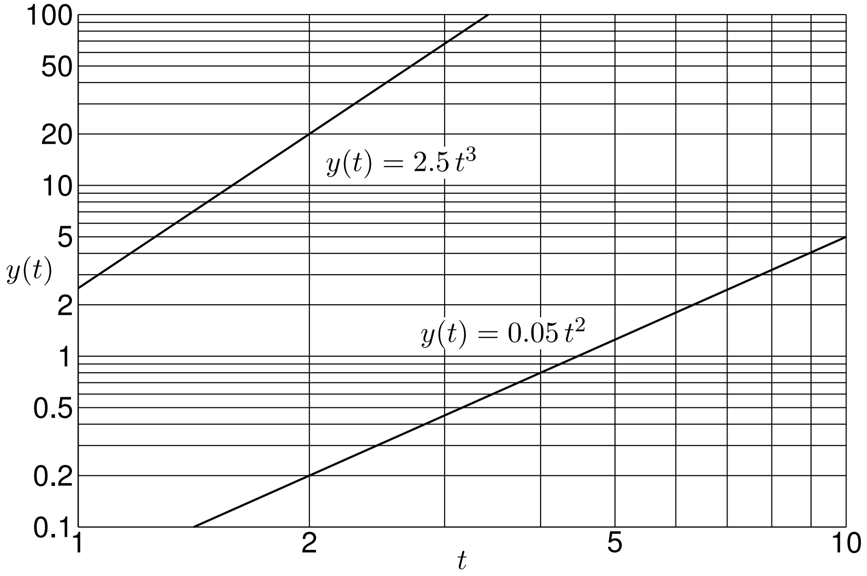


Figure 92: A graph drawn using the pic language `[Loglog.m4]`.

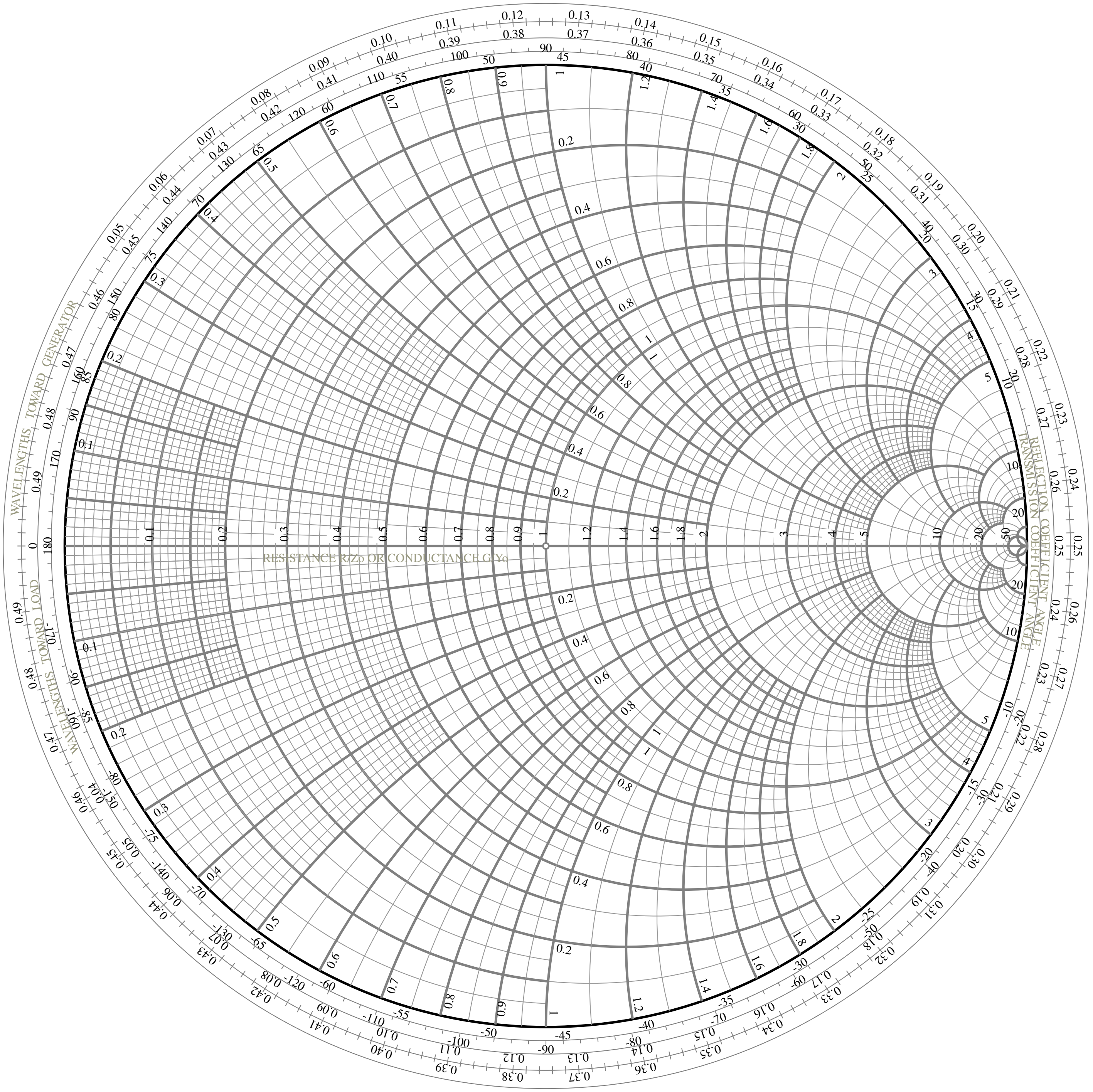
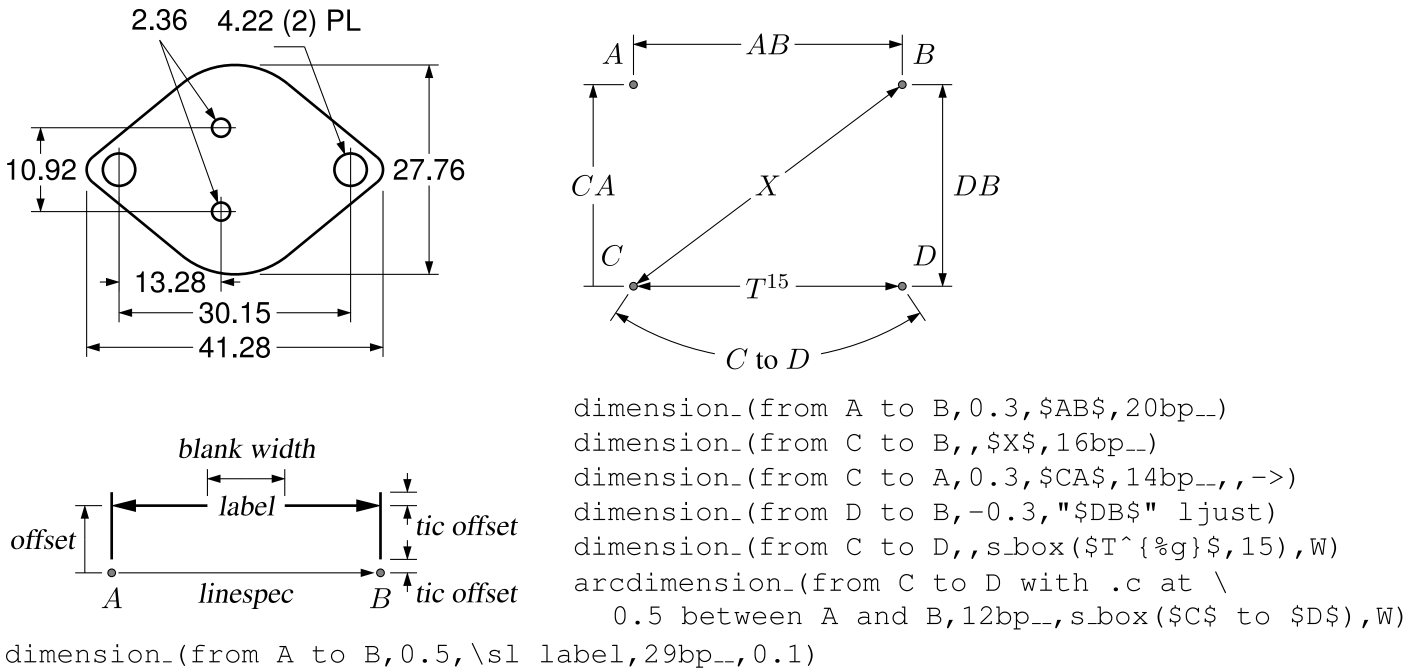

WAVELENGTHS TOWARD GENERATOR
WAVELENGTHS TOWARD LOAD
RESISTANCE R/Zo OR CONDUCTANCE G/Yo
REFLECTION COEFFICIENT ANGLE
TRANSMISSION COEFFICIENT ANGLE

Figure 93: 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 94: 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]`.

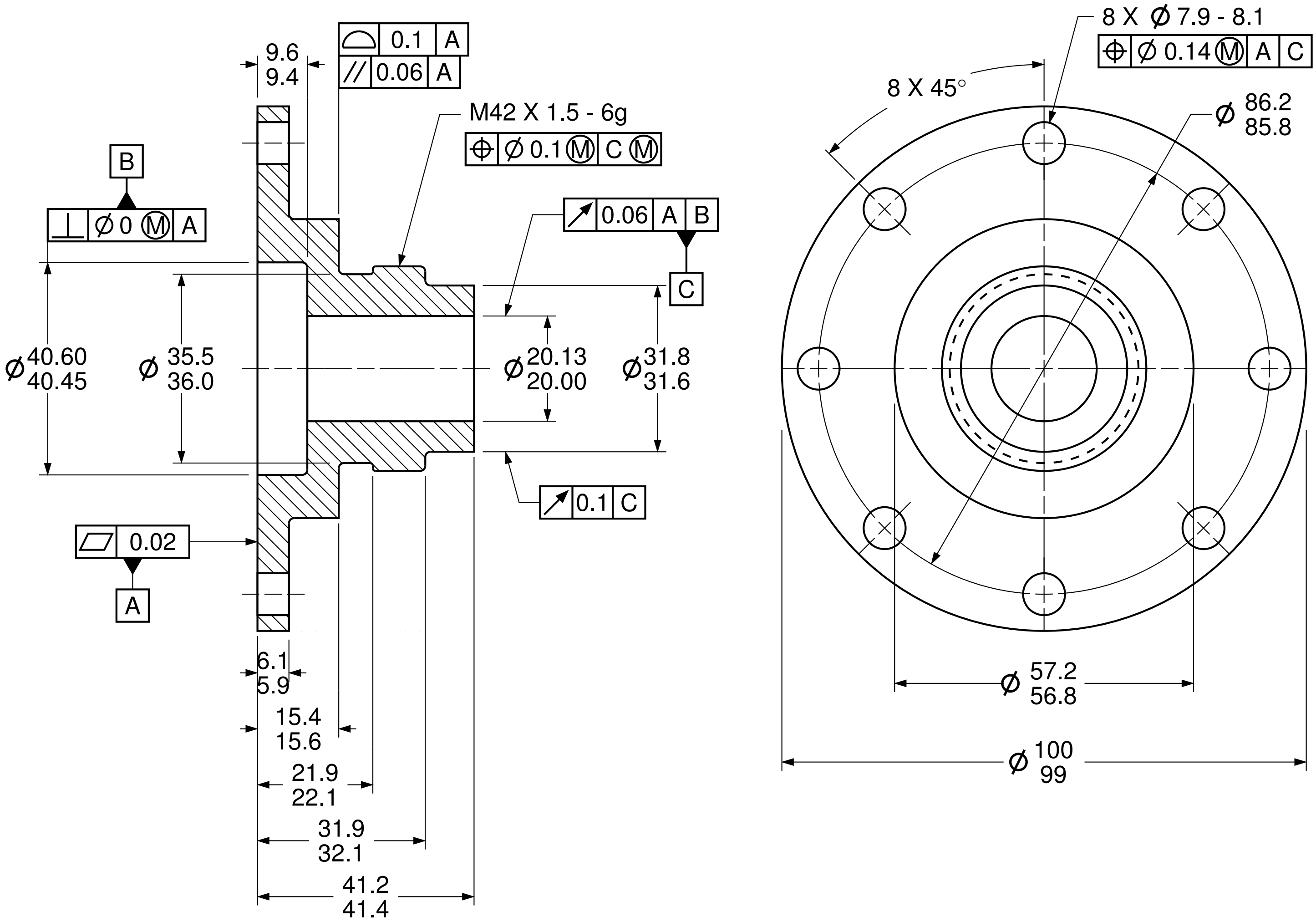


Figure 95: Dimensioning with tolerances [Plate.m4].

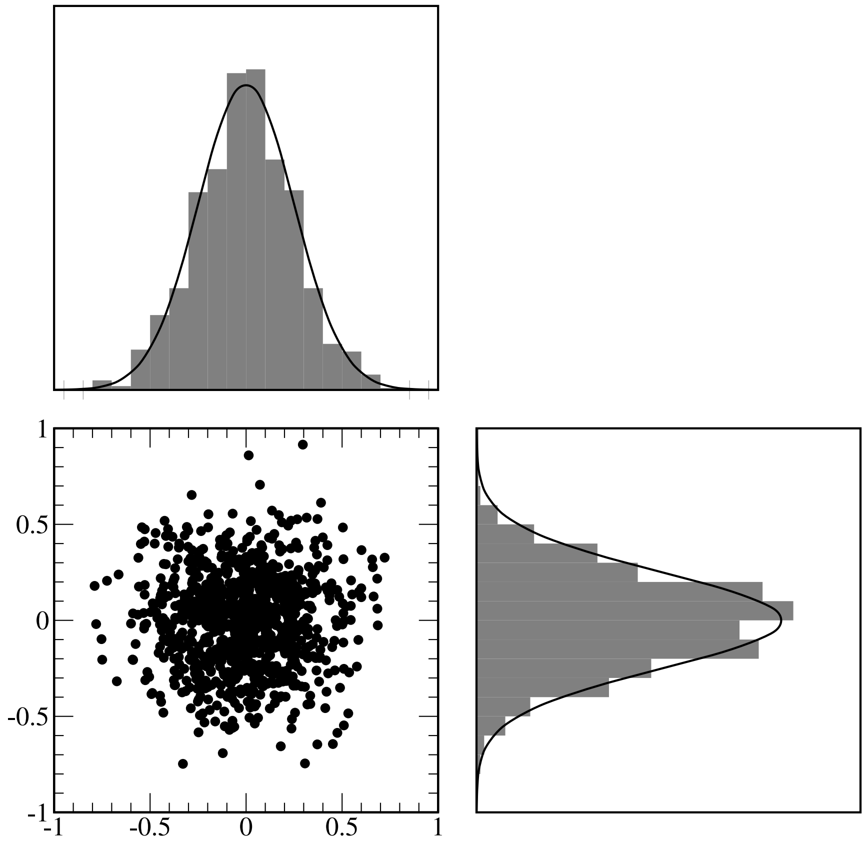


Figure 96: Testing random number generation using dpic macro `randn(`*array name, n, mean, std dev*`)` which calls pic built-in `rand() [random.m4]`.

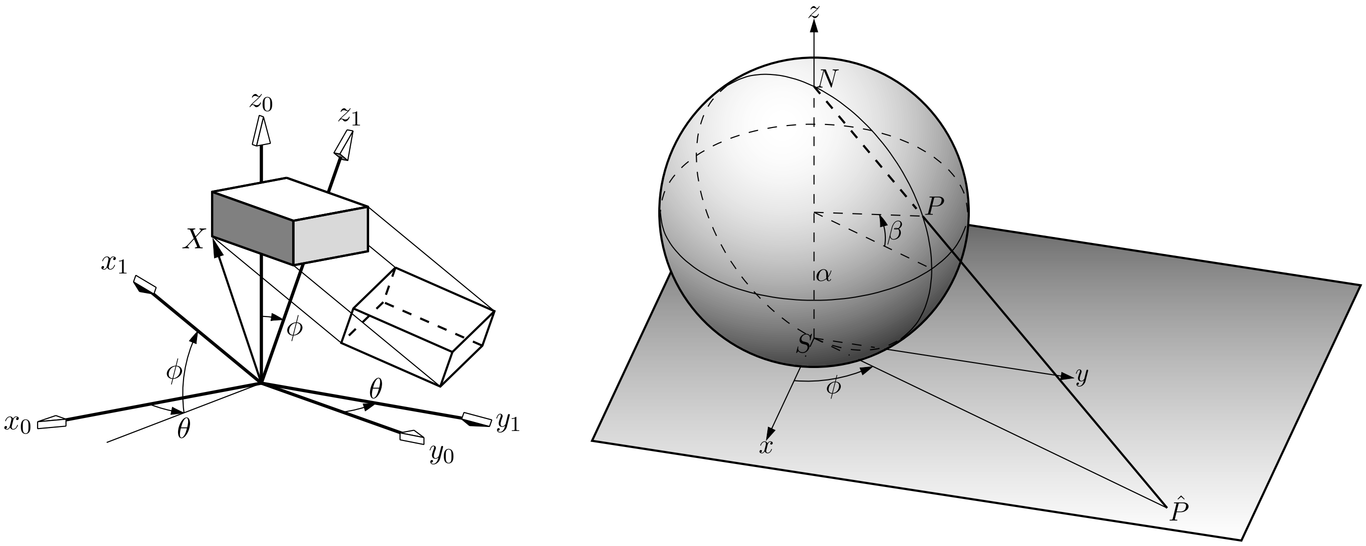


Figure 97: Test of `project` and other `lib3D` macros, showing the projection of a solid onto the y_1, z_1 plane by sighting along the x_1 axis. `[exp.m4]`.

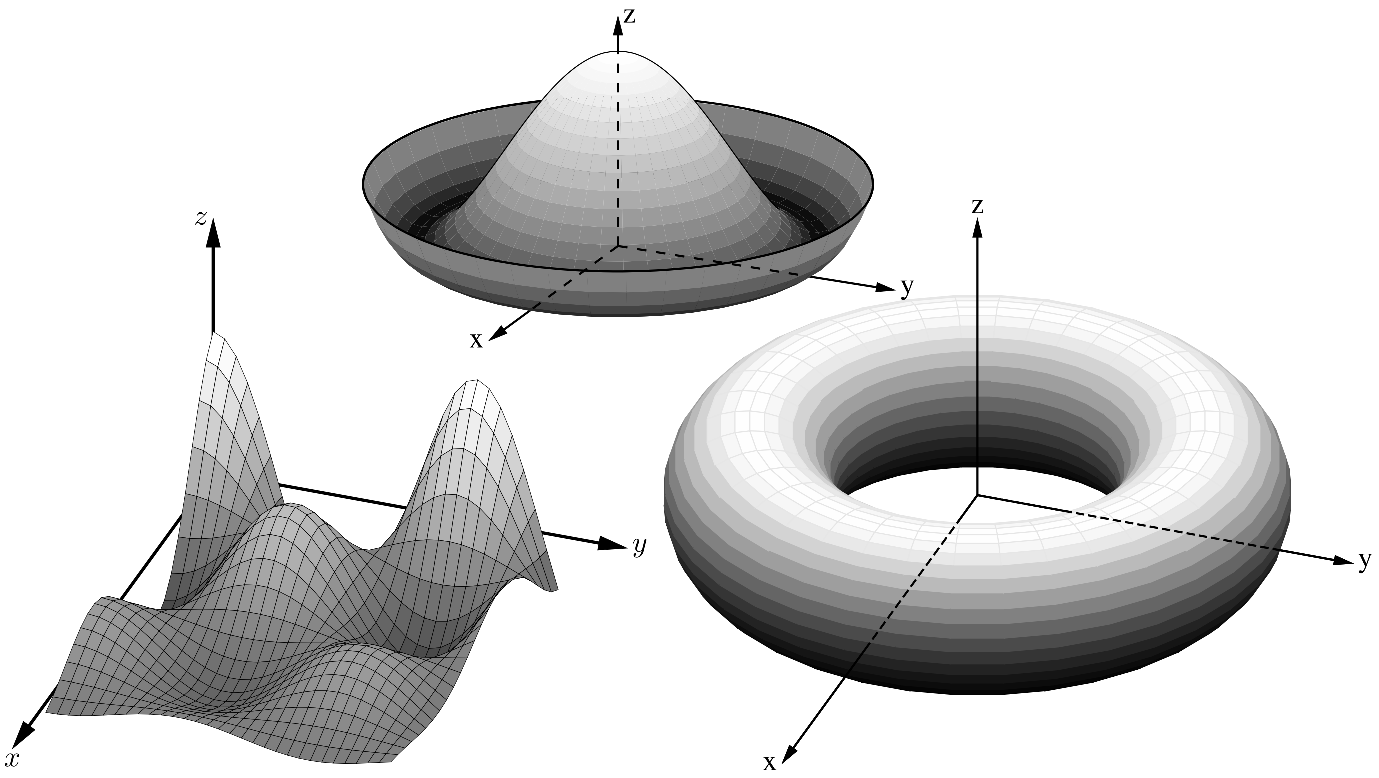


Figure 98: Plotting surfaces using gray scales `[graysurf.m4]`.

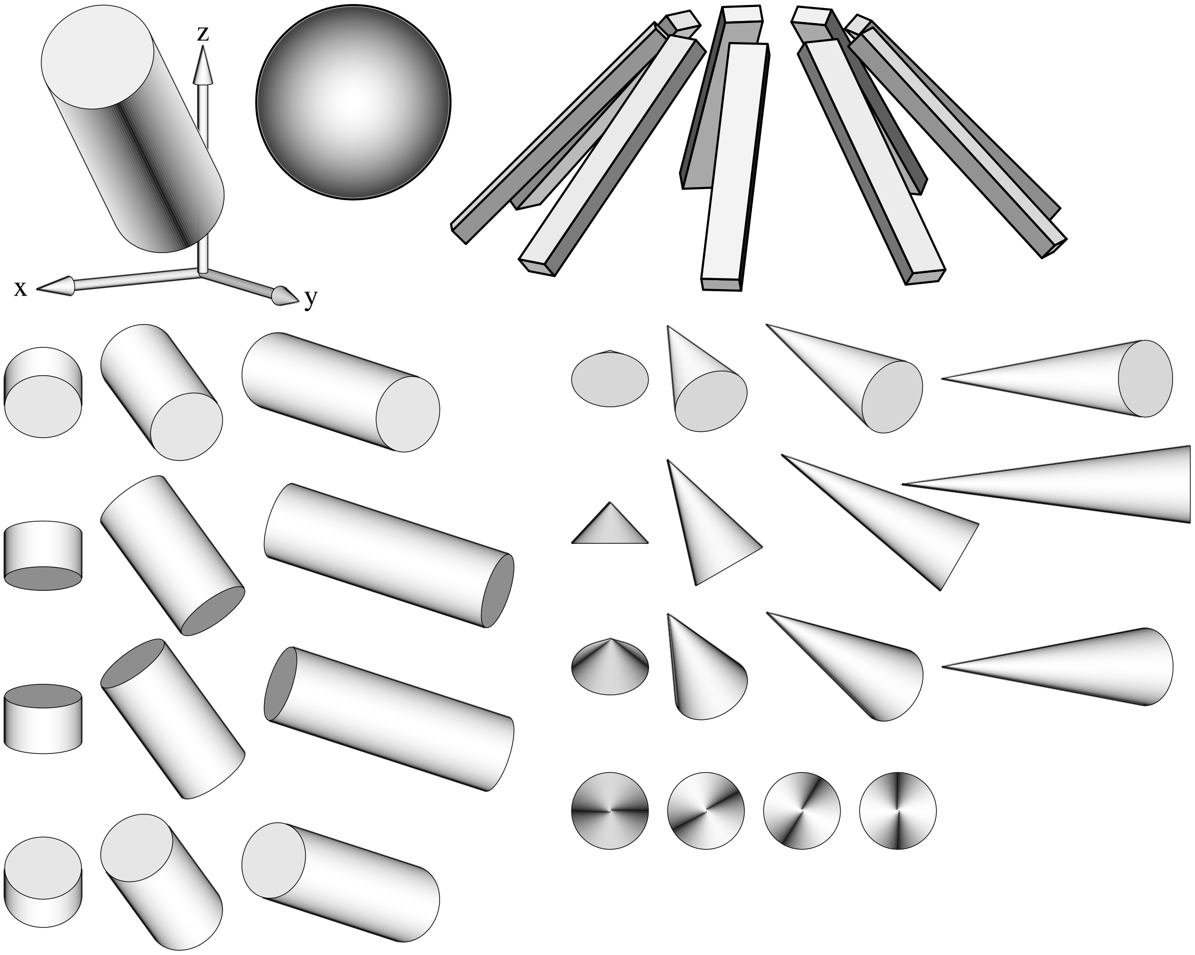


Figure 99: Basic shapes [shapes.m4].

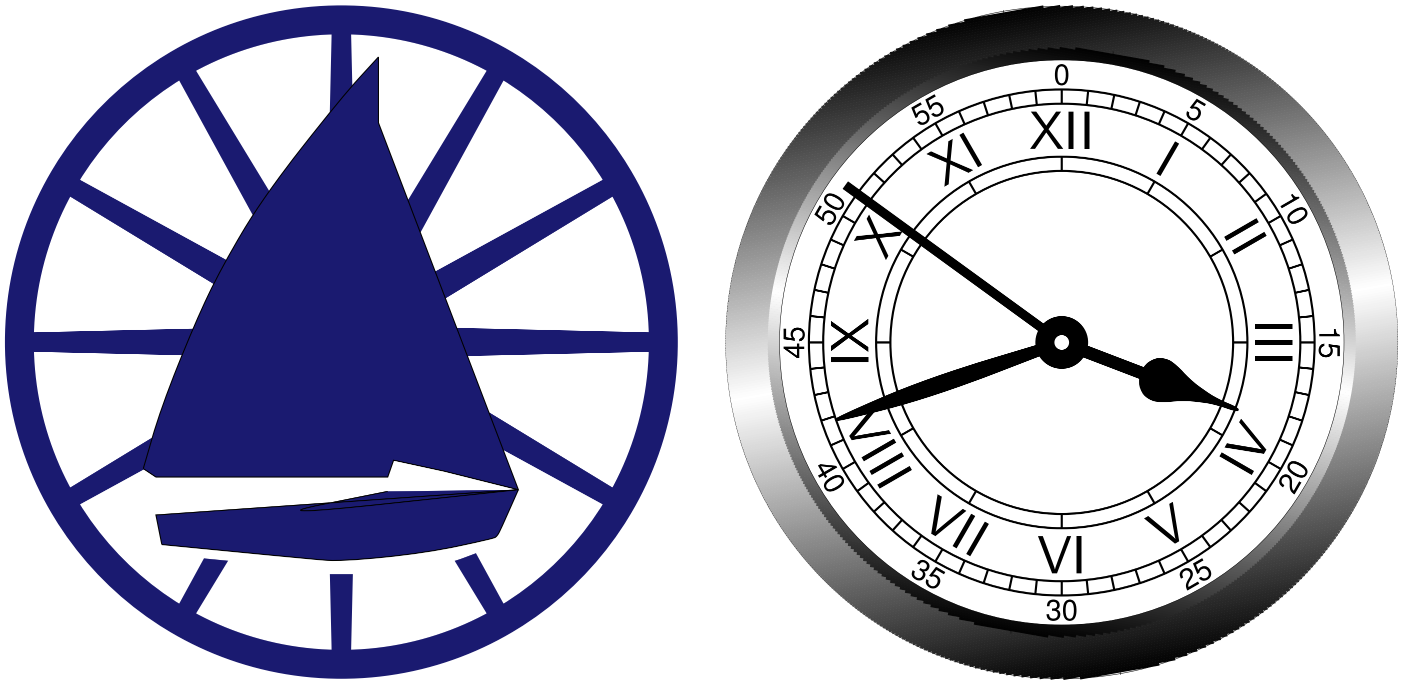


Figure 100: Conestoga Sailing Club (illustrating the filling of arbitrary shapes) and an antique clock face with shading and rotated text `[csc.m4]`.

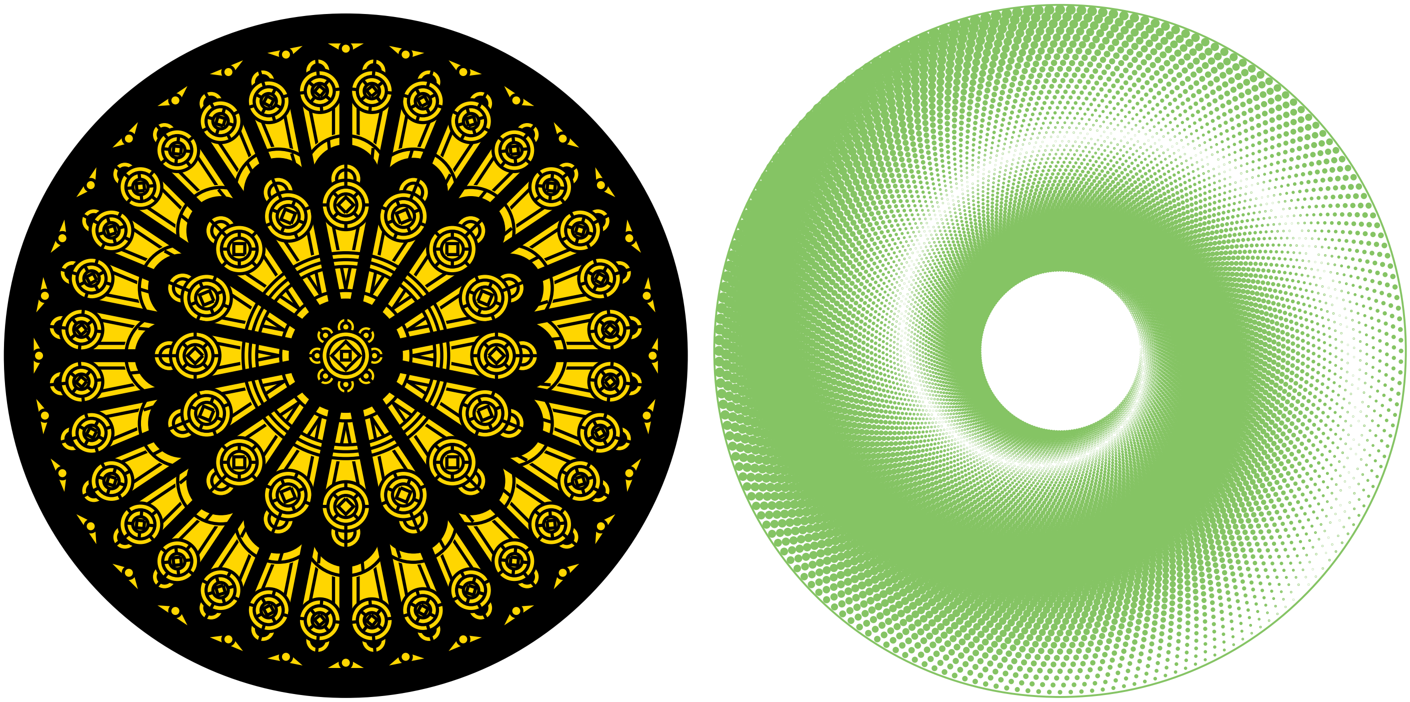


Figure 101: The left object, used for testing `dipic,` is redrawn from a detail of the set design for the musical *Dracula.* This consumes much LaTeX main memory but can be produced directly as pdf using `dpic -d`, as svg using `dpic -v`, or as postscript using `dpic -r` since no text formatting is required. The right object adjusts the size of dots to produce a halftone effect `[rose.m4]`.

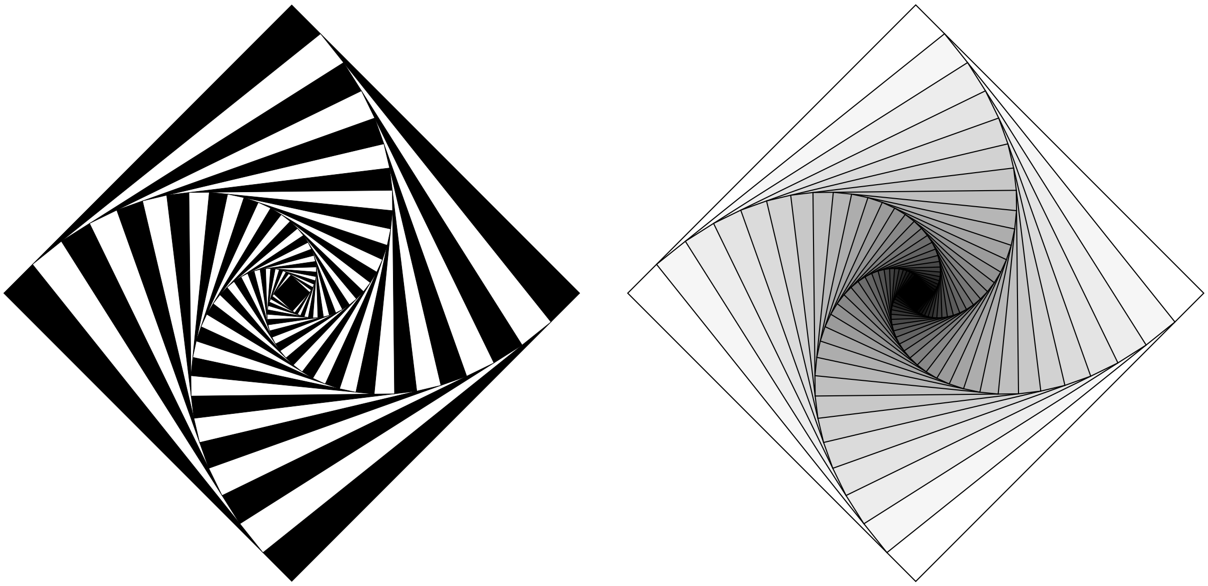


Figure 102: Variations on M. Goossens, S. Rahtz, and F. Mittelbach, *The LaTeX Graphics Companion,* Addison-Wesley 1997, pp. 57-58 `[diamond.m4]`.

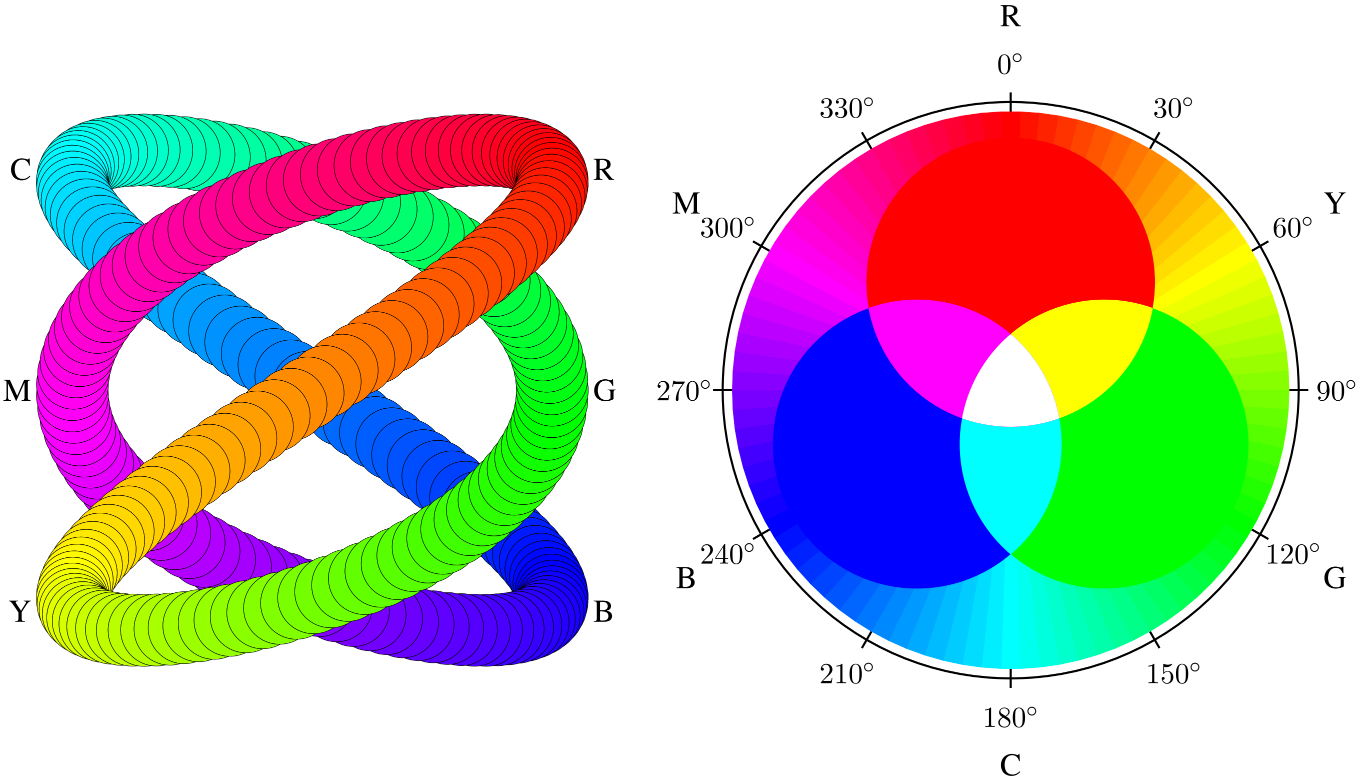


Figure 103: An exercise in calculating RGB colours `[worm.m4]`.

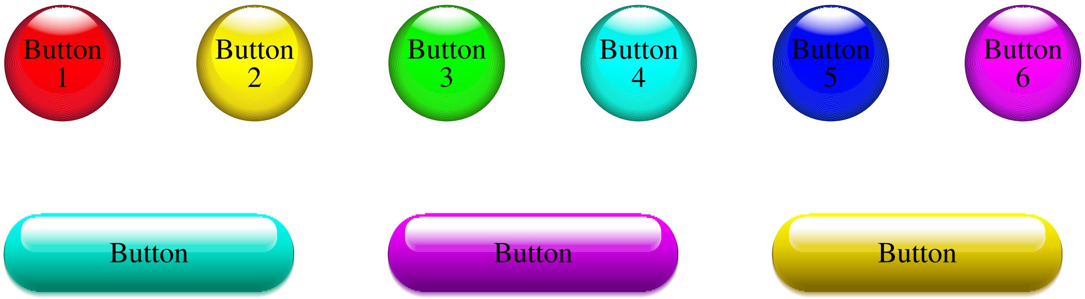


Figure 104: Shading in color [Buttons.m4].

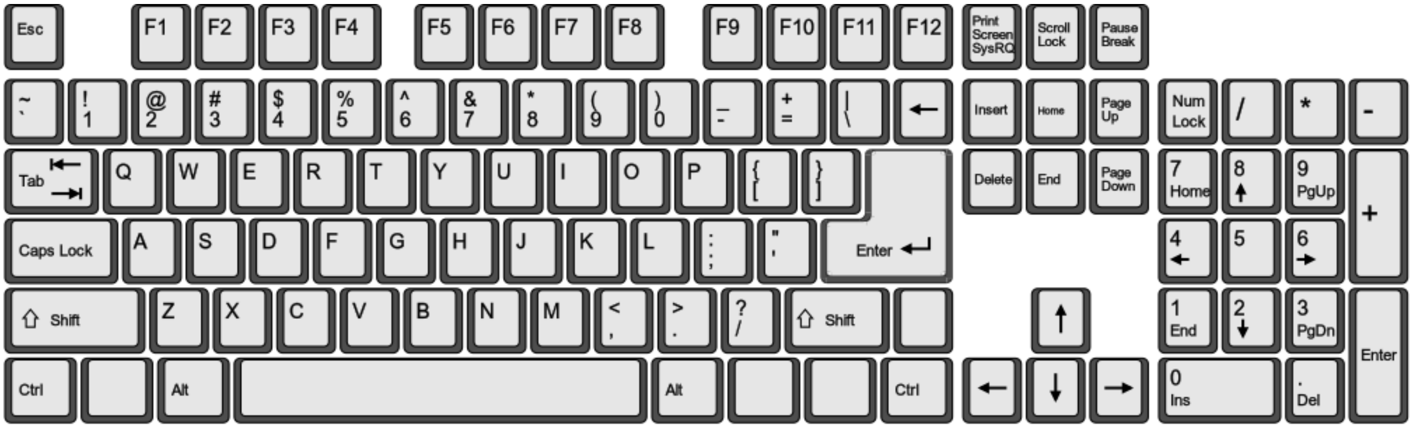


Figure 105: This diagram has been produced as svg with dpic -v (then converted to pdf for inclusion in examples.pdf) [keyboard.m4].

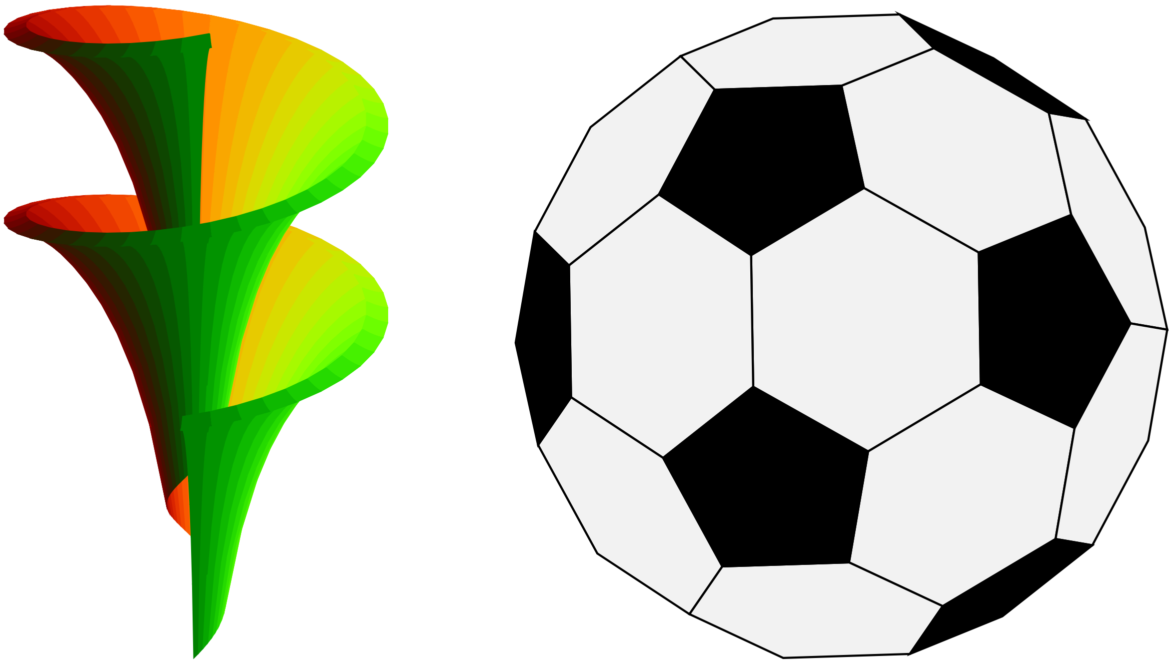


Figure 106: Dini surface and an icosahedron [Dini.m4].

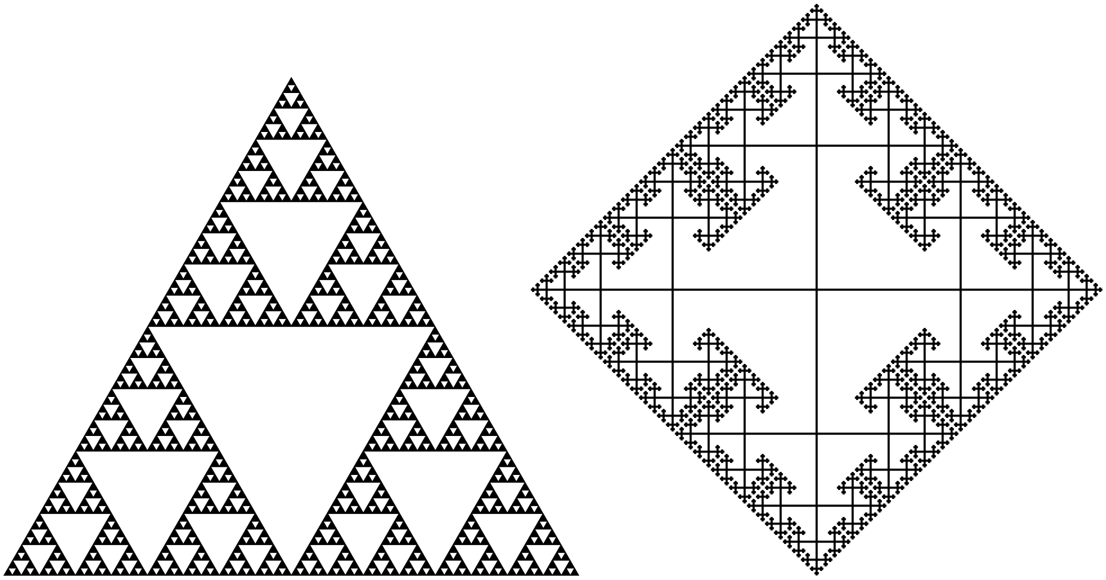


Figure 107: The Sierpinski triangle and a Cayley graph: tests of pic macro recursion [Sierpinski.m4].

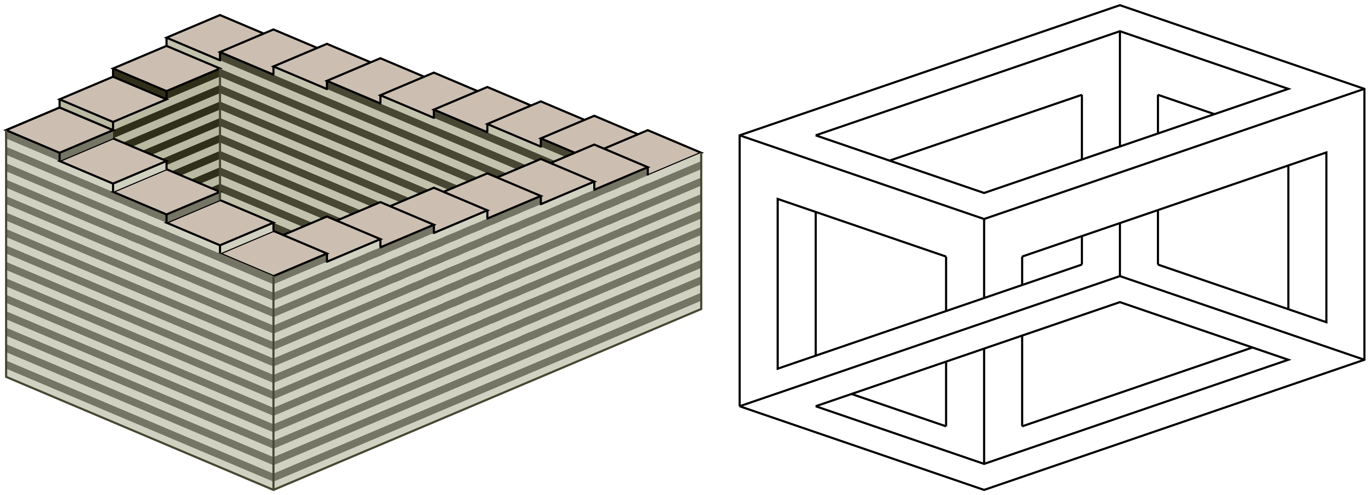


Figure 108: Penrose stairs and an Escher-like object [Escher.m4].

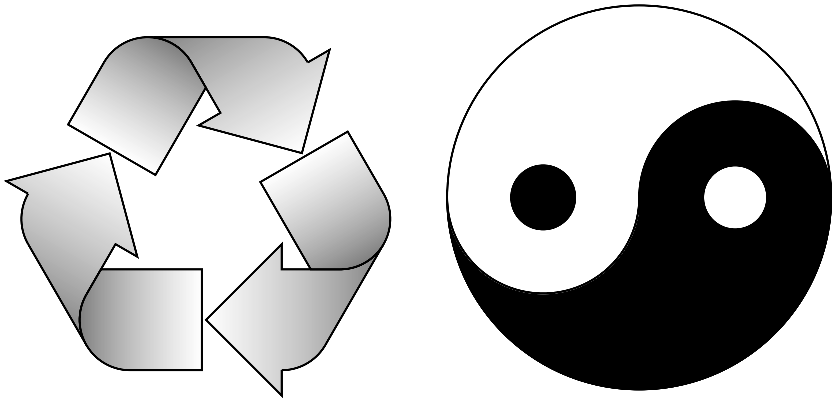


Figure 109: Modest repetition and partial fill [recycle.m4].

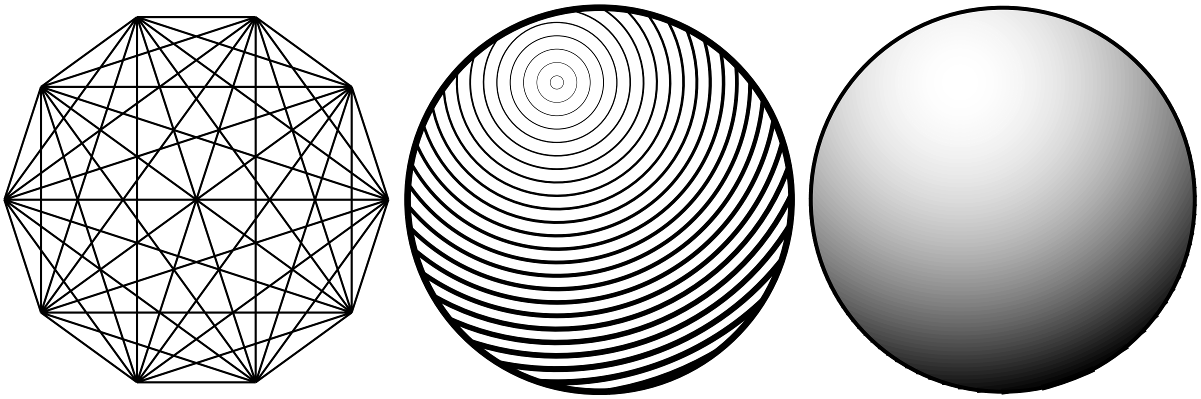


Figure 110: Simple diagrams that are easily drawn by looping [ex15.m4].

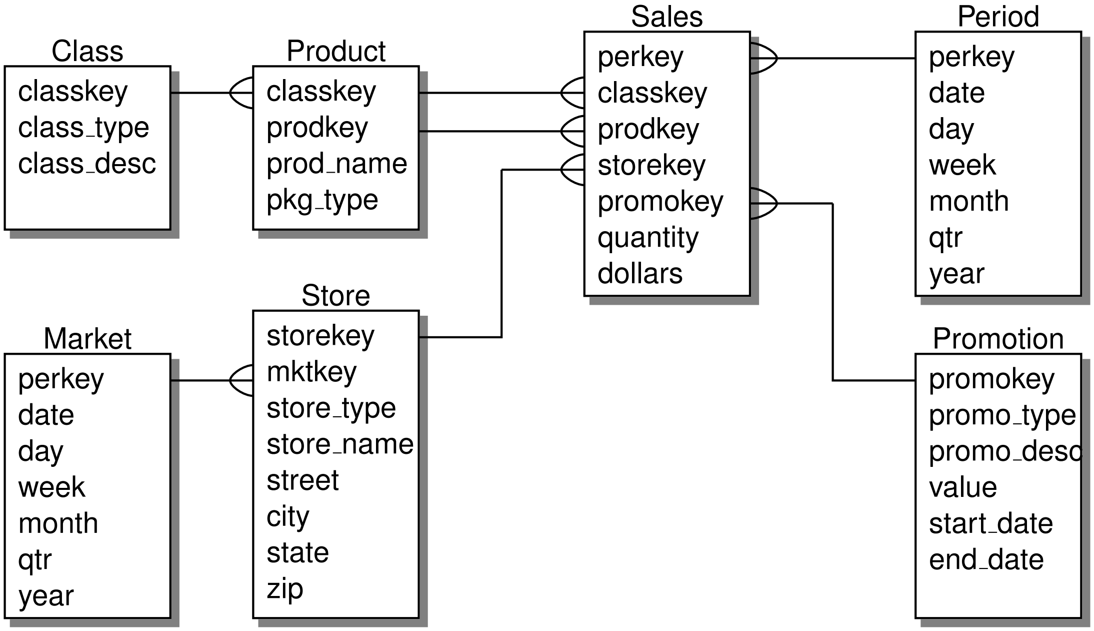


Figure 111: Illustrating shadebox and a custom crowfoot line termination [Crow.m4].

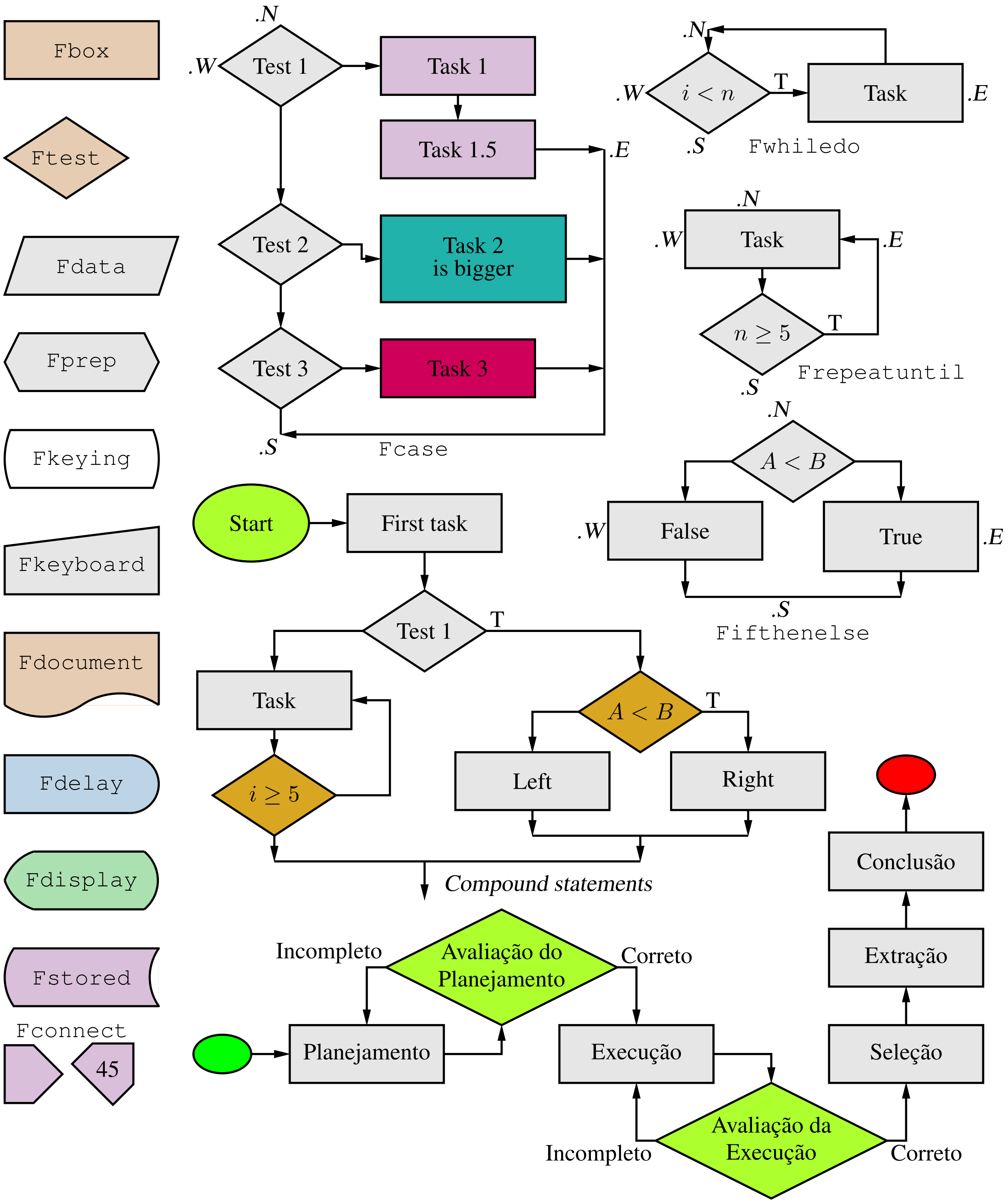


Figure 112: A flowchart sampler `[Flow.m4]`.

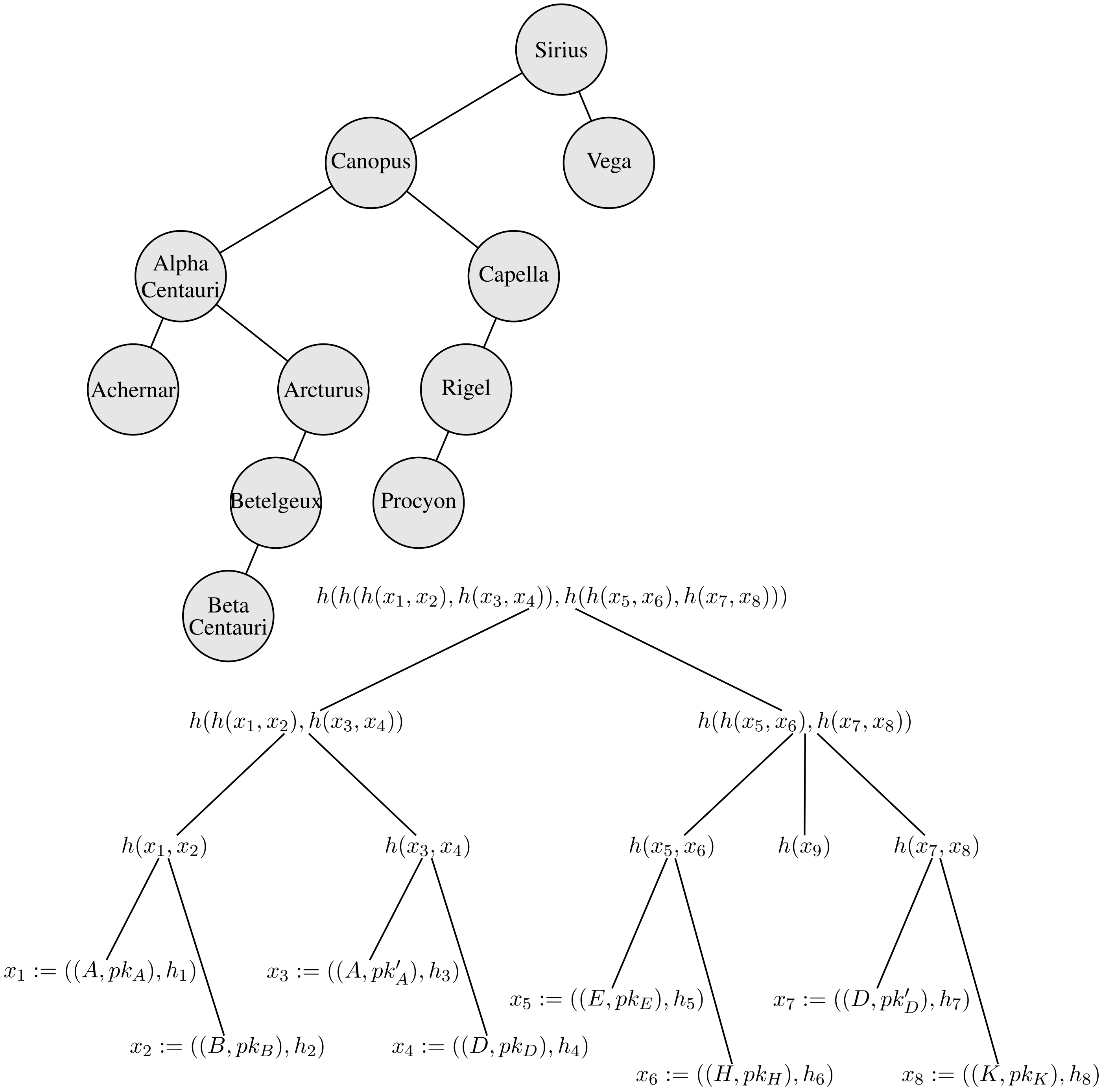


Figure 113: Trees [Btree.m4].

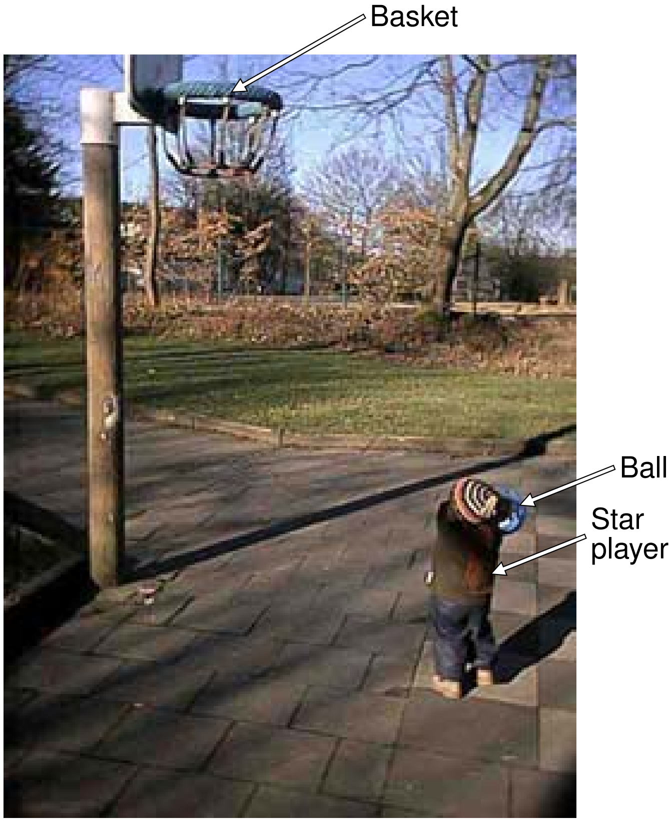


Figure 114: Overlaying a figure with line graphics `[Incleps.m4]`.