Examples: Version 10.8

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. However, 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 are included here in the expectation that you might wish to adapt some of them to your purposes. 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 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.

There are a few files in the examples directory that are not included in this document. To produce .pdf from *file*.m4, for example, type "make *file*.pdf".

Type "make" in the extras directory to see more.
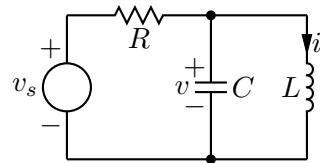
Figure 1: The quick-start example from the manual [quick.m4].
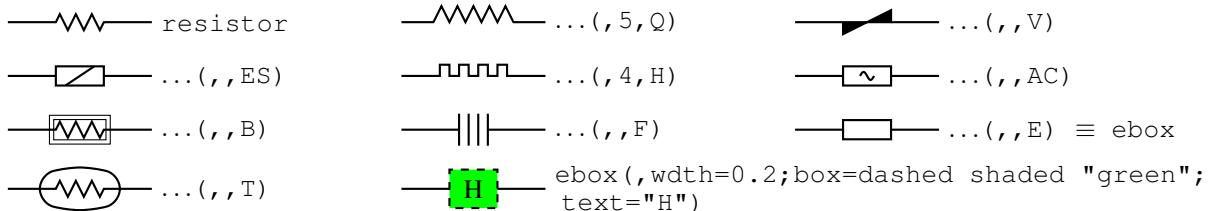
Figure 2: Resistors, showing some variations and the ebox [Resistors.m4].
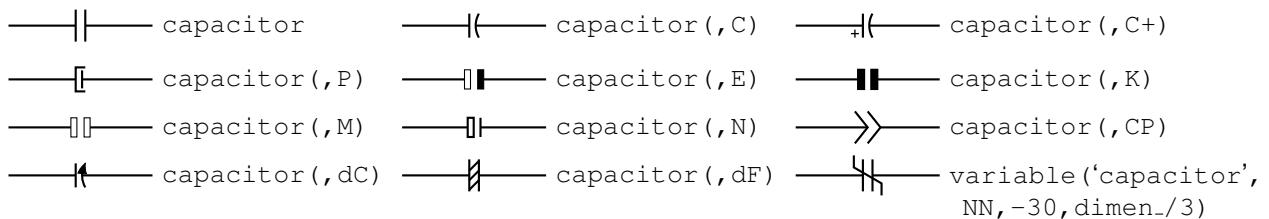

Figure 3: Capacitors [Capacitors.m4].

inductor inductor(,W) inductor(,L)

...(,,3,M) ...(,W,6,P1);
variable ...(,L,9,K)

...(,,,M3) ebox(,,,0) tapped(`ebox(,,,0)',
-,1/4,-dimen_/6,
3/4,-dimen_/6)

addtaps(-,1/2,-dimen_/6) *Tap1* *Tap2*

shielded(
`inductor(,,,M)') shielded(`ebox(,,,0)',R)
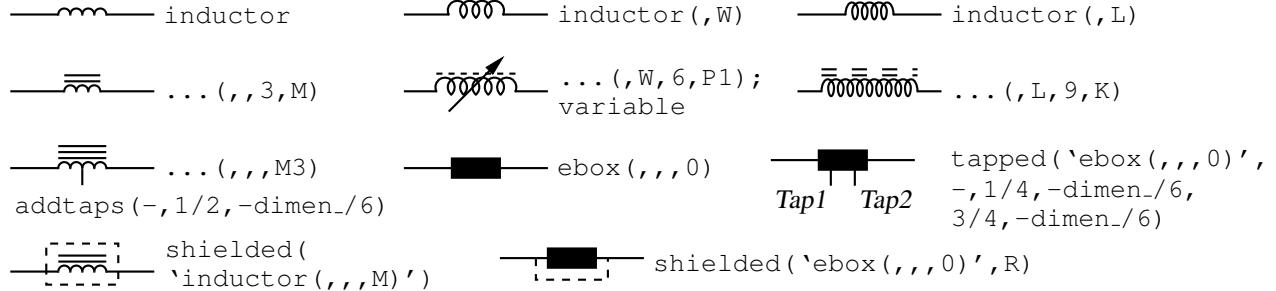
Figure 4: Inductors [Inductors.m4].

diode diode(,z)
diode(,S) diode(,K) diode(,Z,RE)
diode(,V) diode(,ZK)
diode(,v) diode(,CR) diode(,U)
diode(,w) diode(,L) diode(,T)
diode(,B) diode(,F)
diode(,b) diode(,Sh) diode(,P)
diode(,G) diode(,M) diode(,LE)
diode(,D) diode(,MK) diode(,LER)
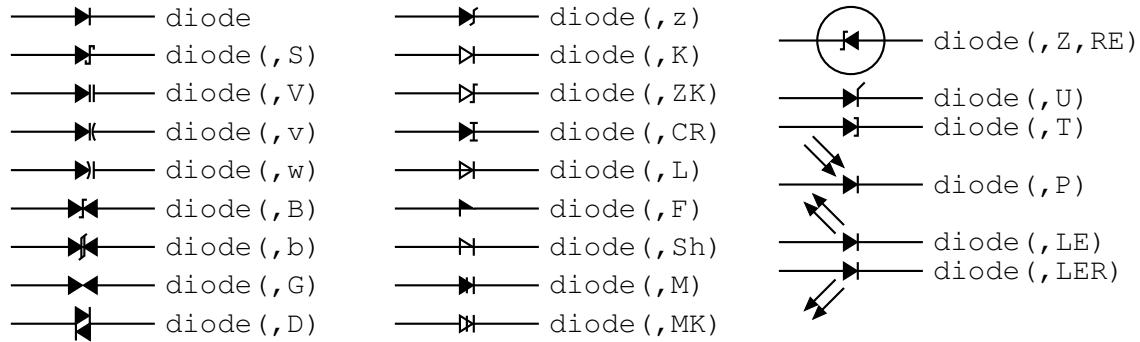
Figure 5: Diodes: appending a K to the second argument draws an open arrowhead [Diodes.m4].

Head *A2*
A1 *Tail*
em_arrows() ...(ND,45) ...(I) ...(ID) ...(E) ...(NT)
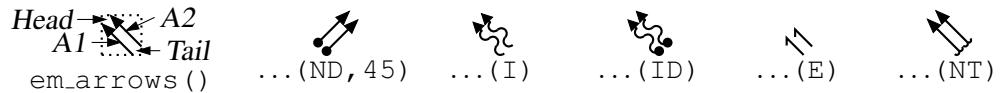
Figure 6: Radiation arrows [Emarrows.m4].

C S

A

P

L

N

A N uN NN uNN
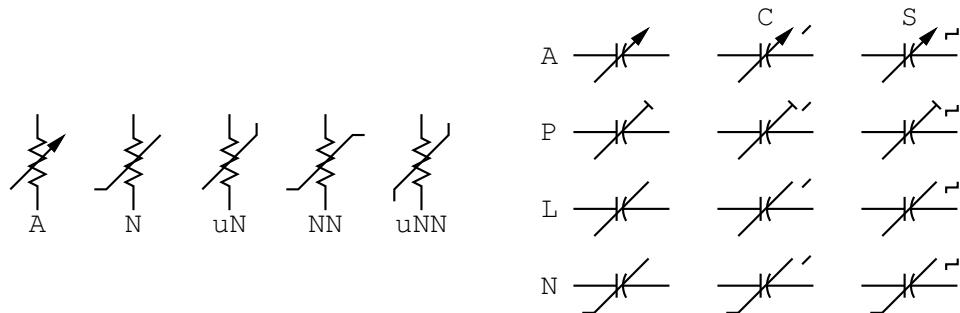
Figure 7: Arrows and marks for showing variability [Variable.m4].
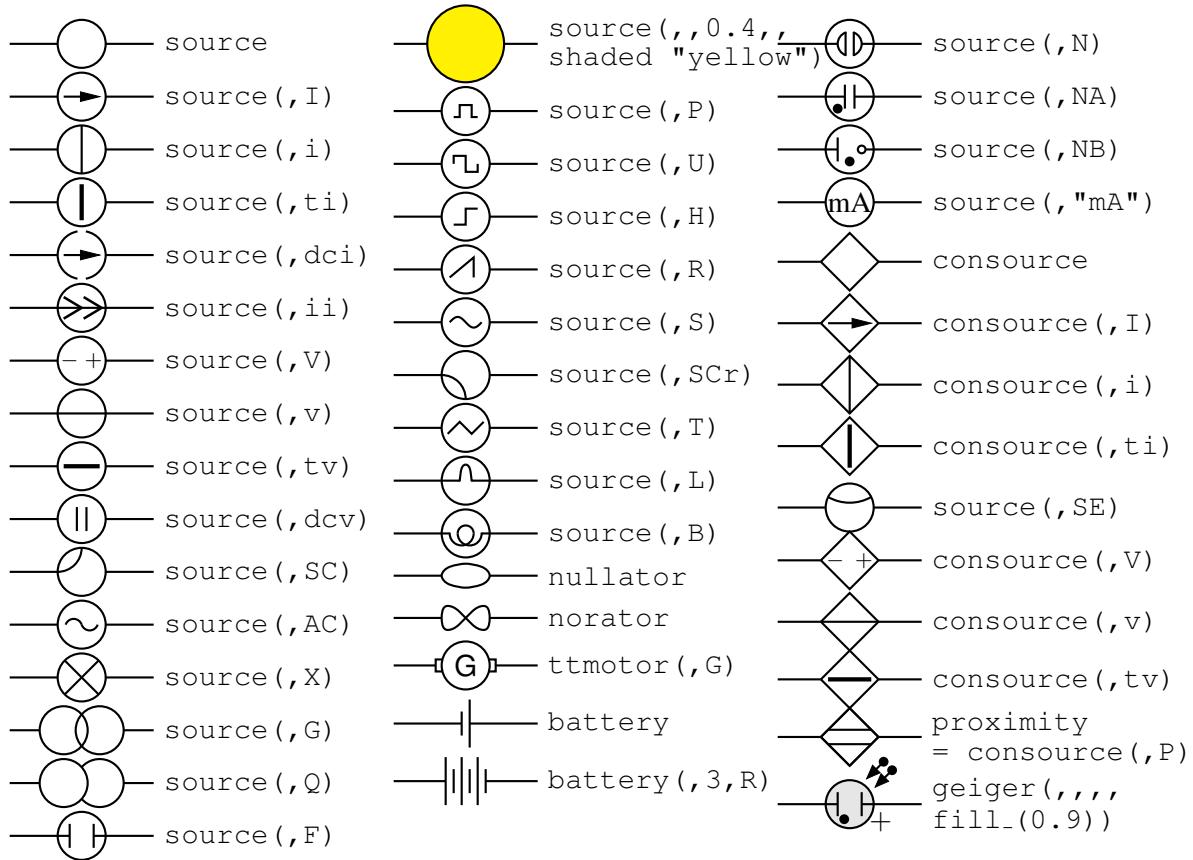
Figure 8: Sources and source-like elements [Sources.m4].
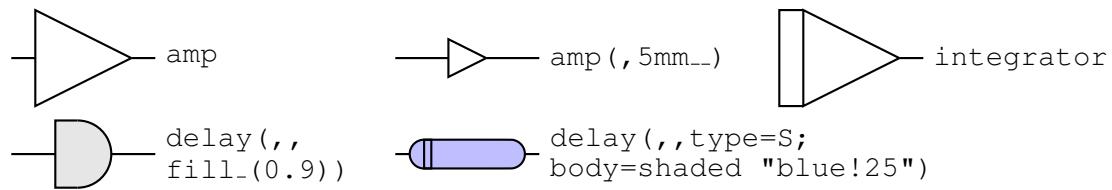
Figure 9: Macros amp, delay, and integrator [AmpTable.m4].
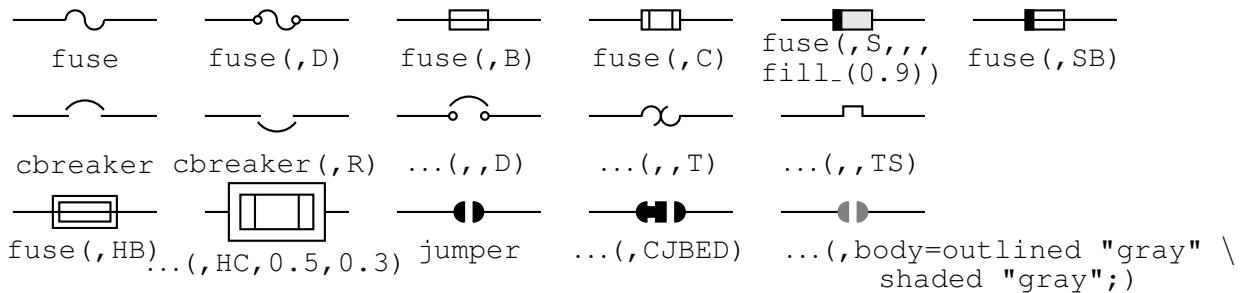
Figure 10: Macros fuse, cbreaker, and jumper [Fuses.m4].
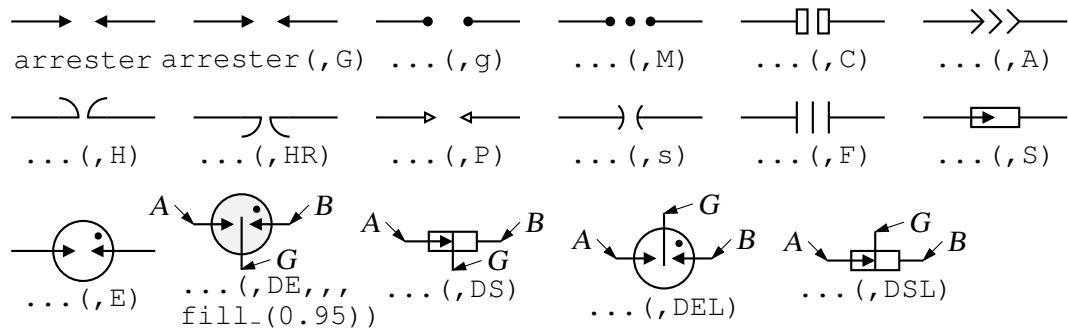
Figure 11: The `arrester` macro [`Arresters.m4`].
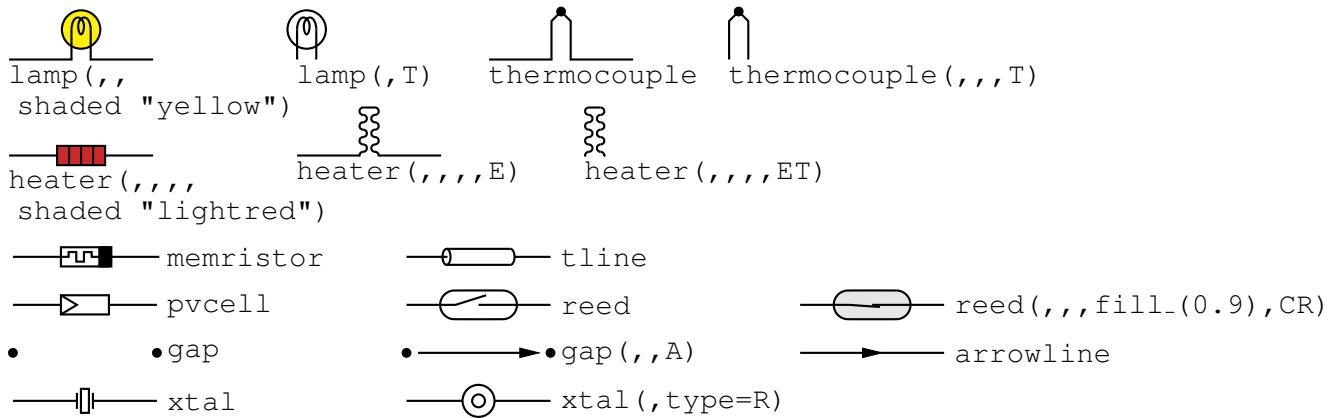
Figure 12: Additional two-terminal elements [`MoreTable.m4`].
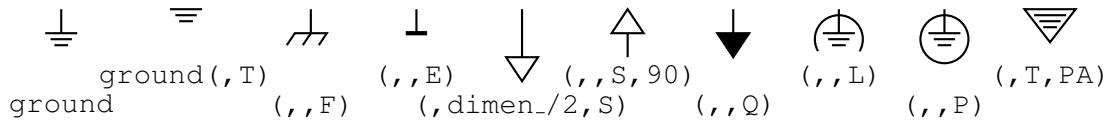

Figure 13: Ground symbols [`Grounds.m4`].
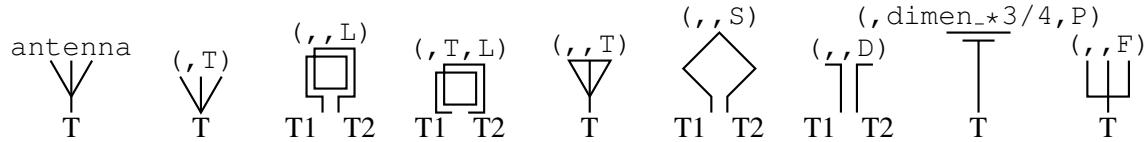

Figure 14: Antenna symbols [`Antennas.m4`].

4

lswitch	(,,O)	(,,C)	(,,DA)	(,,dDO)	(,,uDC)
(,,K)	(,,KD)	(,,KOD)	(,,KCD)	bswitch	(,,C)
dswitch(,,)	(,,WdBK)	(,,WBmdDK)	(,,WBKTr)	(,,WdBL)	(,,WBSd)
(,,WBCb)	(,,WBcCb)	(,,WBcDI)	(,,WBKCo)	(,,WBFDI)	(,,WBFSd)
(,,WBMMR)	(,,WBMM)	(,,WBMR)	(,,WBEL)	(,,WBLE)	(,,WBoKEL)
(,,WBTh)	(,,WBKC)	(,,WBM)	(,,WBCO)	(,,WBMP)	(,,WBoKCP)
(,,WBCY)	(,,WBCZ)	(,,WBCE)	(,,WBRH)	(,,WBRdH)	(,,WBRHH)
(,,WBPr)	(,,WBPrT)	(,,WBPrM)	(,,WBDIGX, text=W)	(,,WBDIGC, circle="M")	(,,WBcDIGC, circle="M")
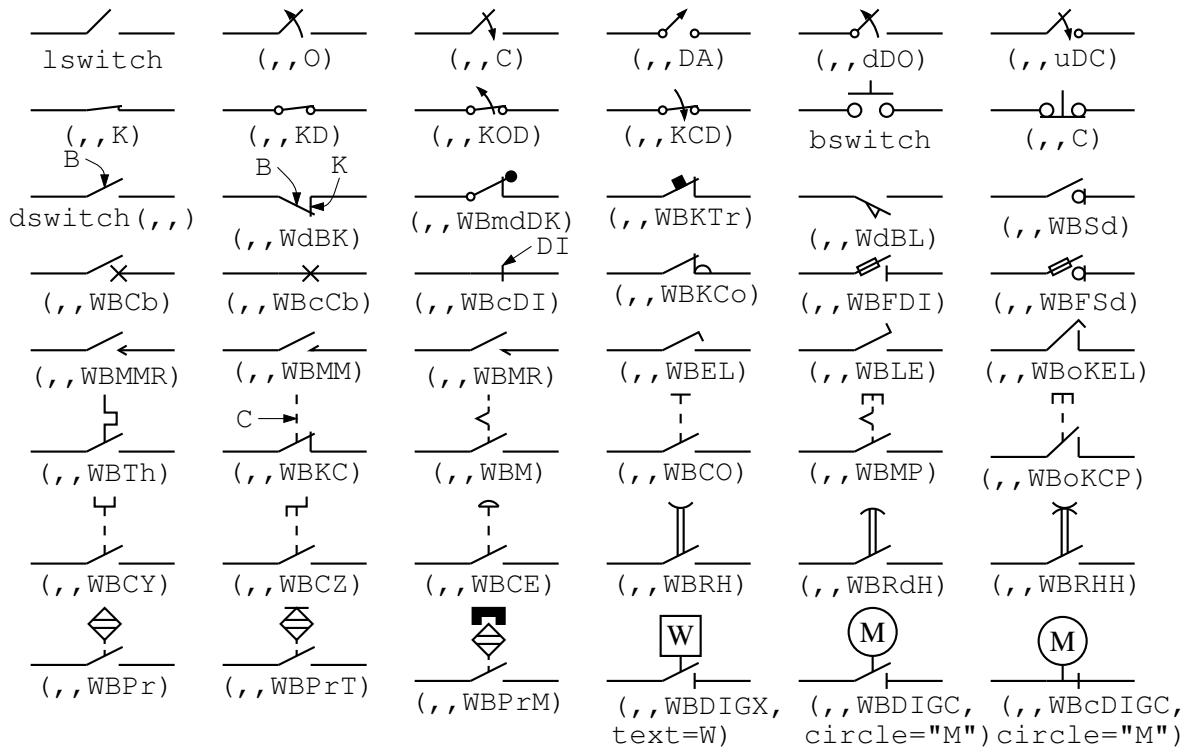
Figure 15: The switch macros; switch(,,,L|B|D) is a wrapper for lswitch, bswitch, and dswitch [Switches.m4].

opamp(,,,
body=shaded "yellow")

Point_(90);
opamp(,,,,PR)

opamp(,,,
body=shaded "orange"
,T)
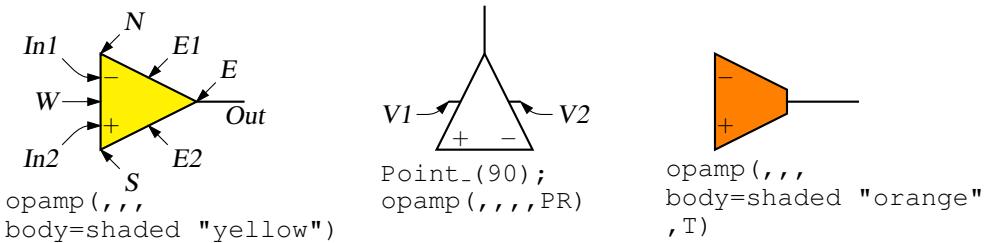
Figure 16: The opamp [Opamp.m4].

speaker
speaker(,,H)

bell

buzzer

buzzer(,,C)

microphone

earphone

earphone(,,C)

microphone(A)
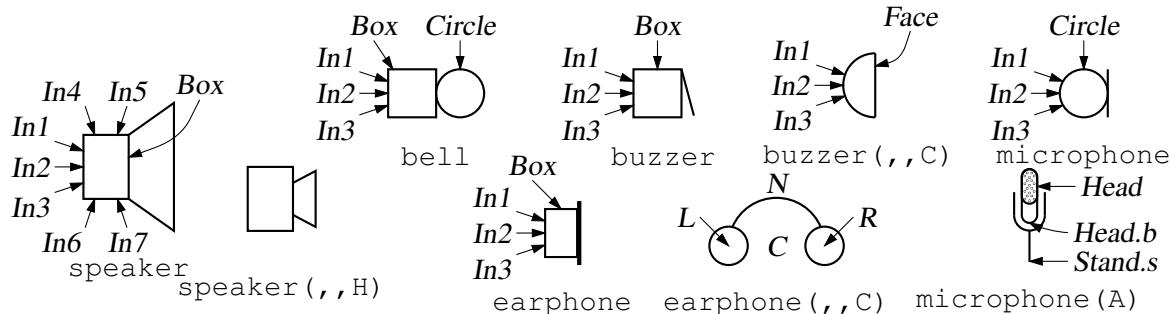
Figure 17: Audio elements [Audio.m4].

5
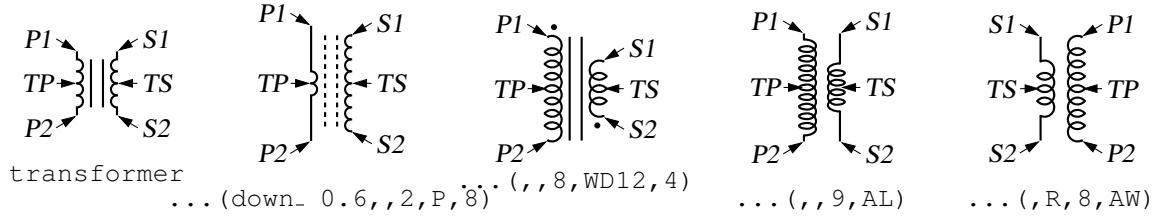
Figure 18: Some variations of the transformer element, drawing direction down [Xform.m4].
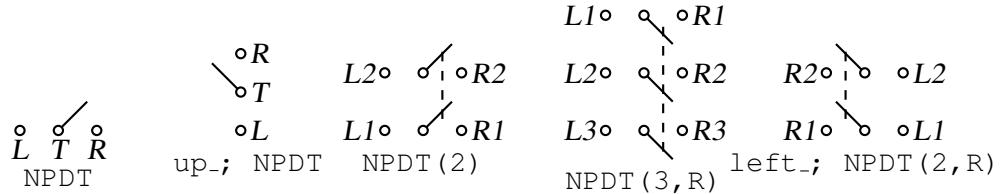
Figure 19: Double throw with the NPDT macro [NPDT.m4].
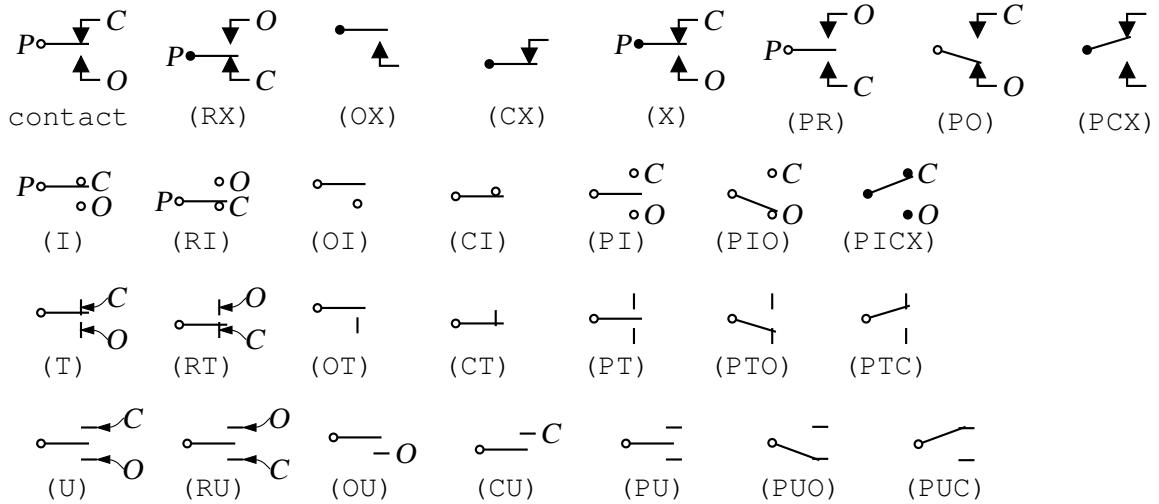
Figure 20: A non-exhaustive sampling of contact macro variations [Contact.m4].
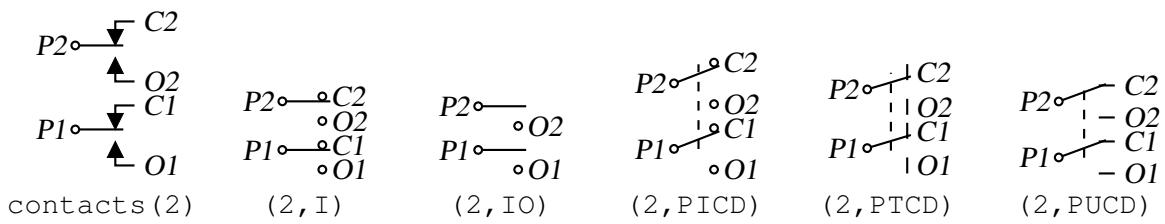

Figure 21: The contacts macro [Contacts.m4].

6
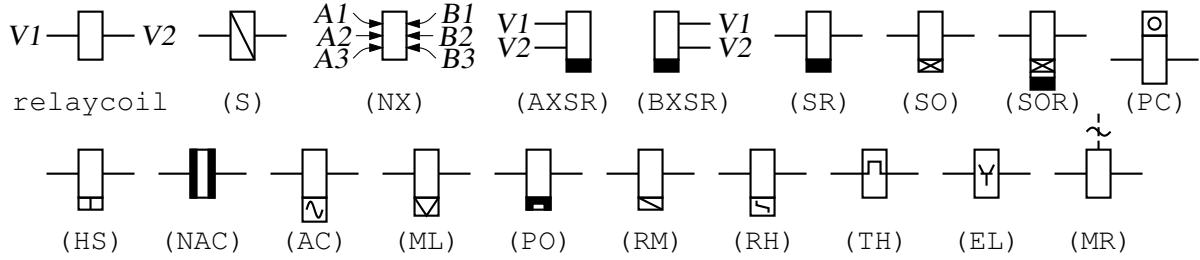
Figure 22: The `relaycoil` macro [relaycoil.m4].
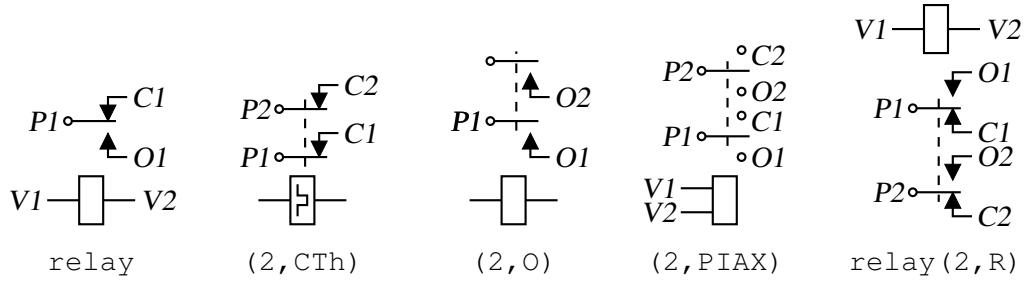
Figure 23: Some variants of `relay` [Relay.m4].
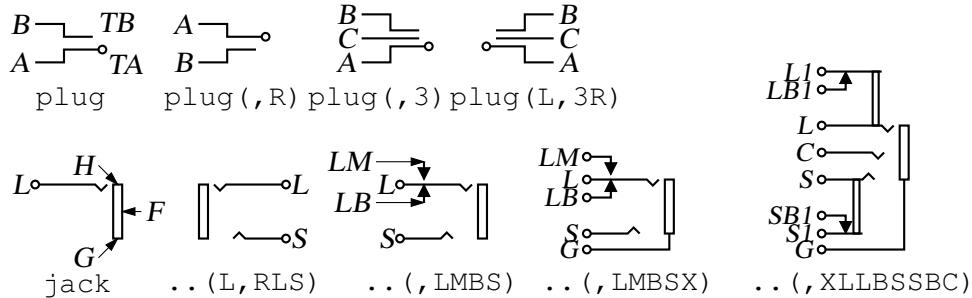

Figure 24: The `jack` and `plug` macros [Jack.m4].
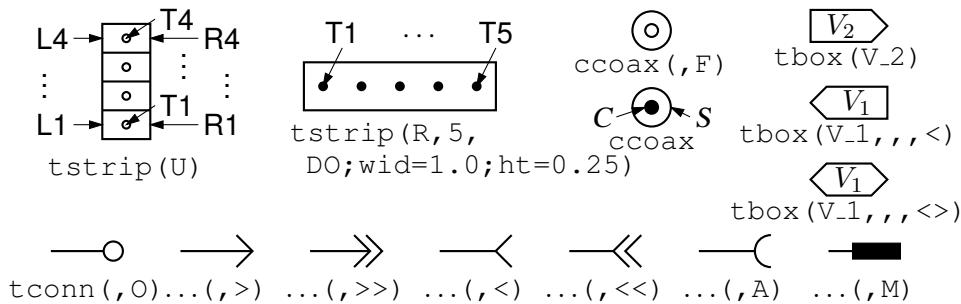
Figure 25: The `tstrip`, `ccoax`, `tconn`, and `tbox` macros [Conn.m4].
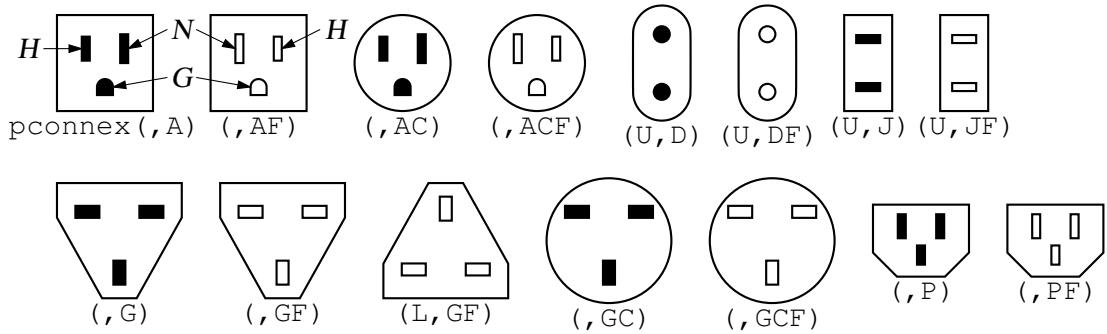
Figure 26: The `pconnex` macro [`Pconn.m4`].

10 Ω 12 Ω 15 Ω 18 Ω 22 Ω 27 Ω 33 Ω 39 Ω 47 Ω 56 Ω 68 Ω 82 Ω 100 Ω
5 %
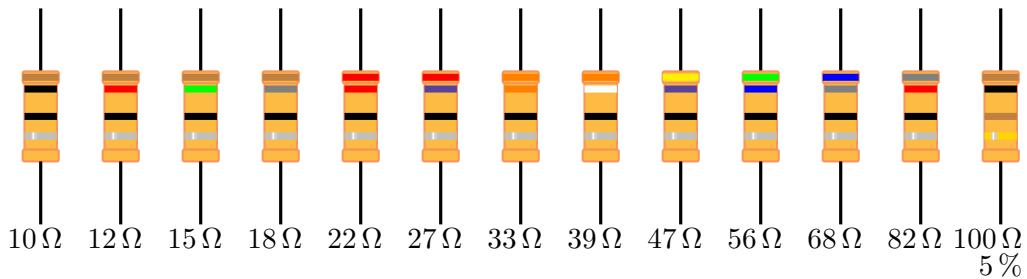
Figure 27: Color-coded through-hole resistors. All are 10 percent tolerance except the last, which illustrates 5 percent [`cbresistor.m4`].

EV_J1772   EV_J3068   EV_CCS1   EV_CCS2   EV_CHAdeMO   EV_NAC
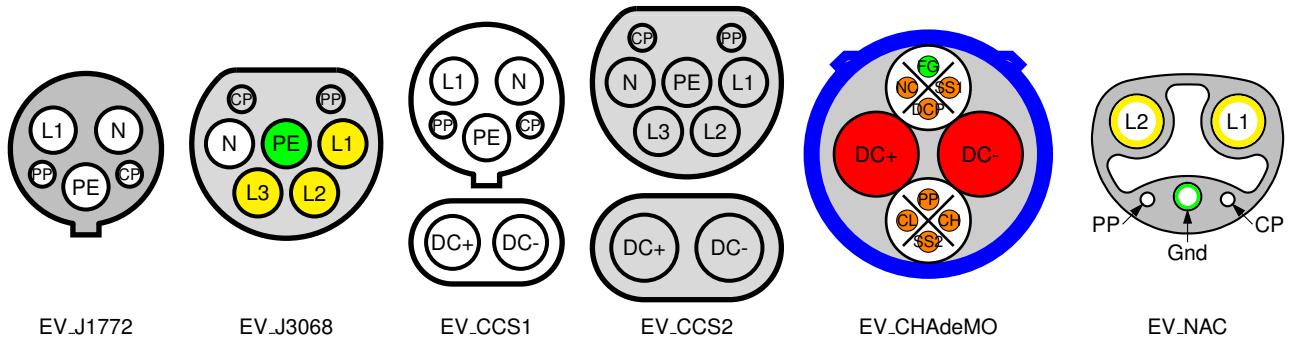
Figure 28: Electric vehicle charging plug patterns make extensive use of *key=value* pairs to set options [`EVplugs.m4`].
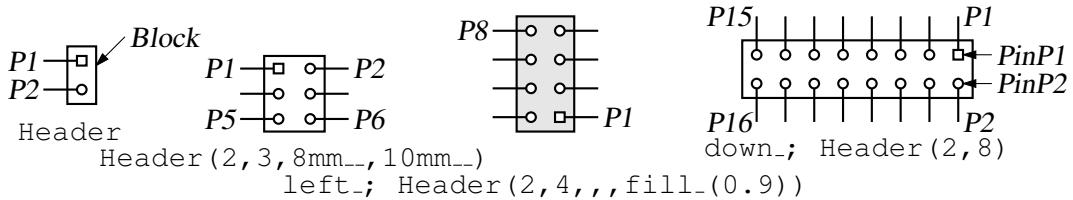
Figure 29: The `Header` macro [`Headers.m4`].
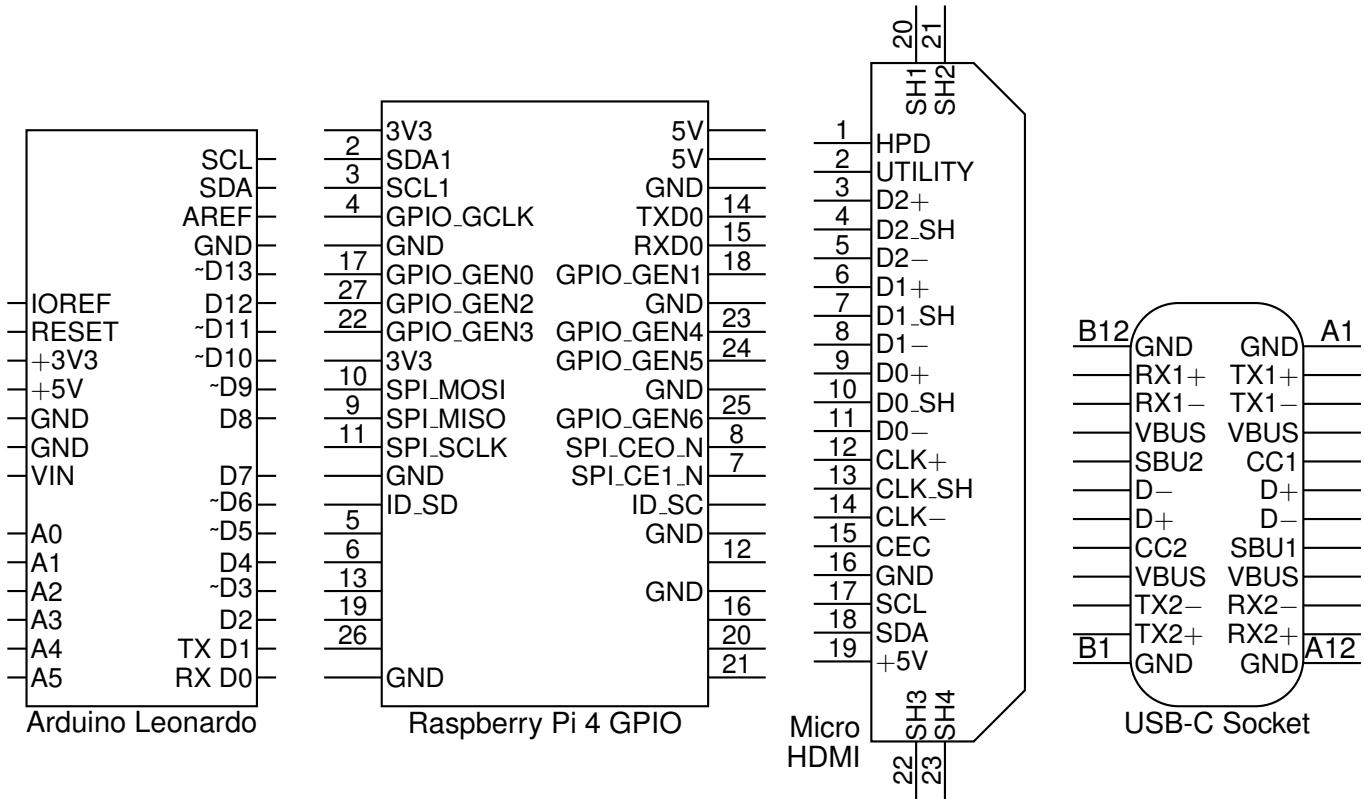
Figure 30: Some connectors with simple geometry and lists of labels [Connectors.m4].
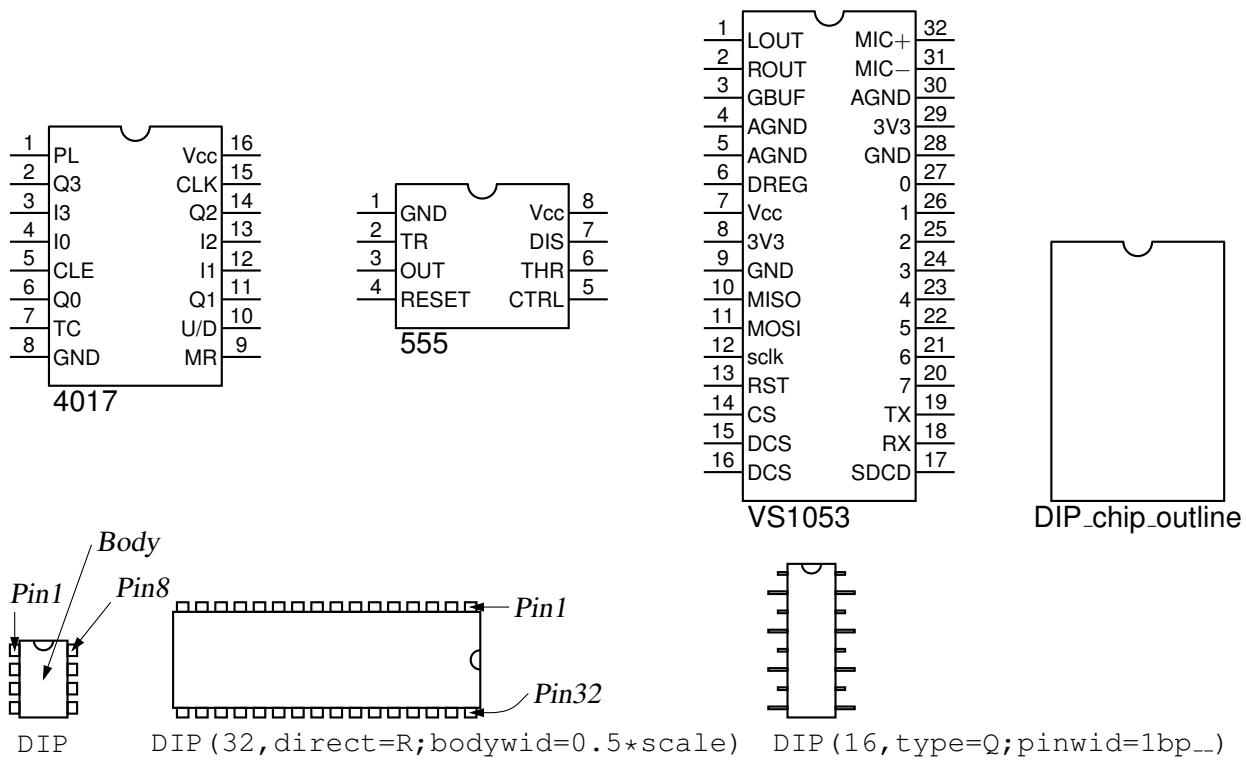
Figure 31: IC package outlines and examples [Chips.m4].

G
S j_fet D

G
S j_fet(right_ dimen_,,P,E) D

G
S e_fet D

G
S e_fet(,,P) D

G
S e_fet(,,P,S) D

G
S c_fet D

G
S c_fet(,,P) D

G
S d_fet D

G
S d_fet(,,P) D

G
S d_fet(,,P,S) D

G
S g_fet D

D
G g_fet(up_ dimen_,,P) S

G0 G1
...(,,dBSDFQuM1)
mosfet(,,dBSDFQM1,E)

G1 G0

dG
F
S mosfet(,,dGSDF,) D

uH
...(,,uHSDF,)

dM
E uB
Q ...(,,dMEDSQuB,)

...(,,uMEDSuB)

G
S B D

G
S Fe_fet D

G
T S Fe_fet(,,TEDSQuB) D

dT
Z ...(,,ZSDFdT,)

D
G IRF4905 S
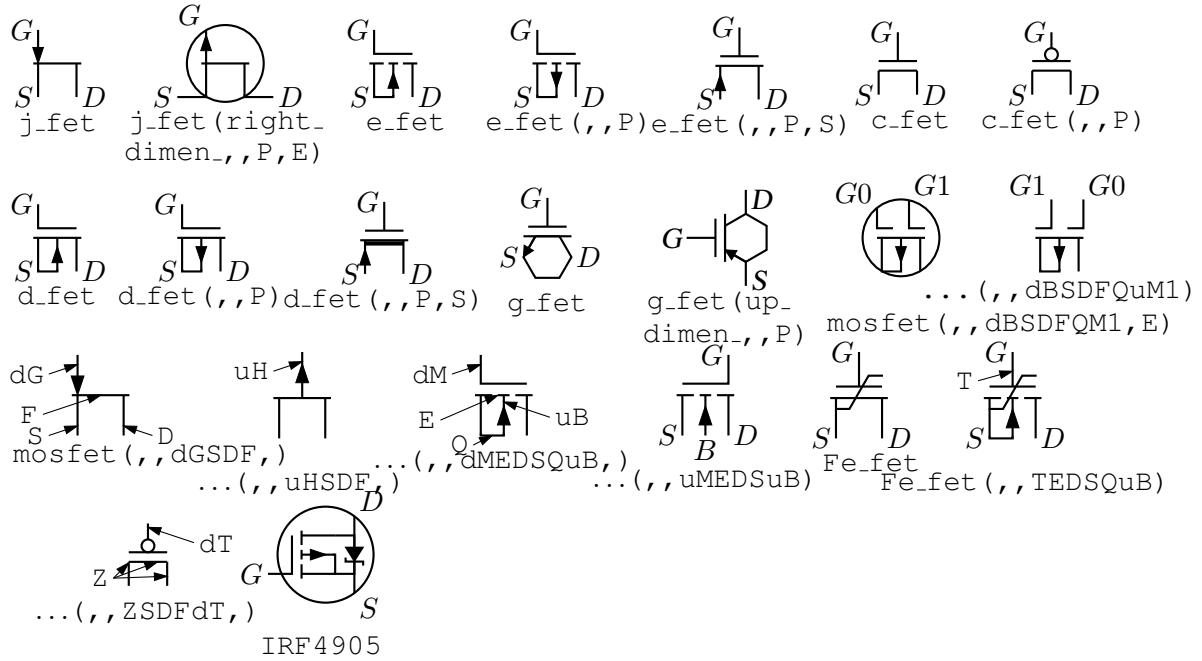
Figure 32: FETs, showing programmable components and example customizations [fet.m4].

E B2
B1
ujt(up_ dimen_,,,E)

E B2
B1
ujt(,,P,)

B2 E
B1
ujt(,R,,)

B2 E
B1
ujt(,R,P,)
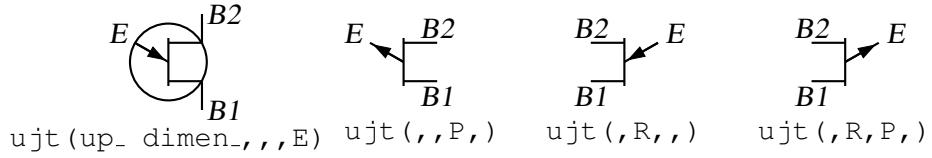
Figure 33: UJT examples [ujt.m4].

A
K thyristor G

T1
T2 ...(,B) G

T1
T2 ...(,BRK) G

T1
T2 ...(,BE) G

A
K .(,A) G

A
K ...(,F) G

T1
G T2 ...(,BRE)

A
G K ...(,UARE)

A
G K ...(,AV)

A
G K ...(,IEC)

A
K ...(,UAH) G
...(,N)

A Ga
K ...(,UANRE)
...(,SCR)

A
K G ...(,SCRE)
...(SCRRE)

A
G K ...(SCS)
...(SCSE)

A Ga
K G ...(,SUSE)
...(SCSE)

A
G K ...(SUSE)

T1
G T2 ...(SBSE)

Q.G
scr(,,Q)

Q2.G
scs(,,Q2)
Q2.Ga

Q3.G
sus(,RE,Q3)

sbs(,E,Q4)
Q4.G
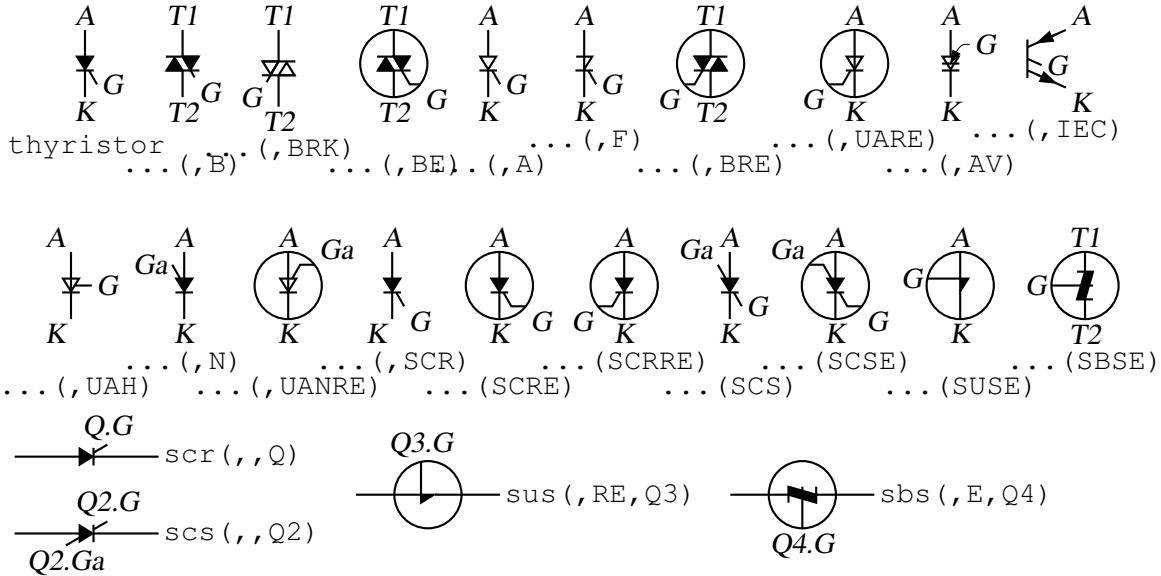
Figure 34: Thyristor examples. The thyristor is a 3- or 4-terminal composite element [thyristor.m4].

10

bi_tr(up_ dimen_)   bi_tr(,,P)   bi_tr(,R)   bi_tr(,,,E)   igbt   igbt(,,LD)
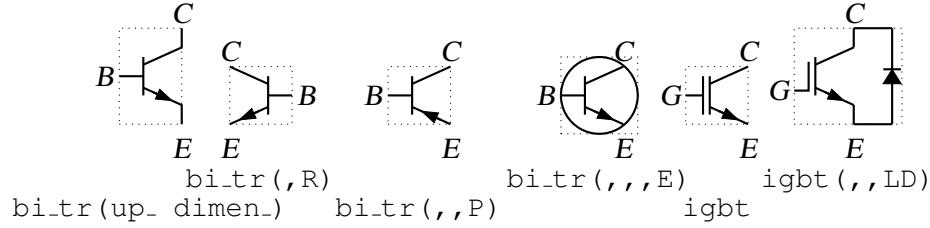
Figure 35: Bipolar transistors (drawing direction: up) [Bip.m4].
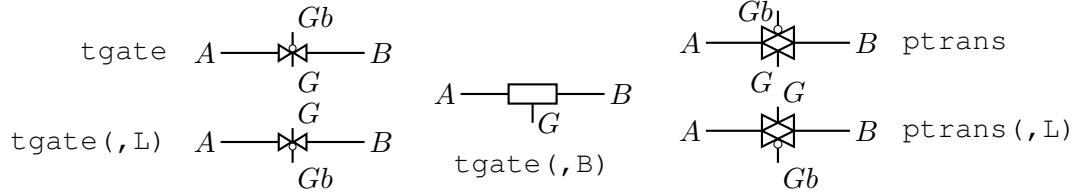
Figure 36: The tgate and ptrans elements [Tgate.m4].

nport(wid 2.0 ht 1 fill_(0.9) "n-port",1,2,3,4)
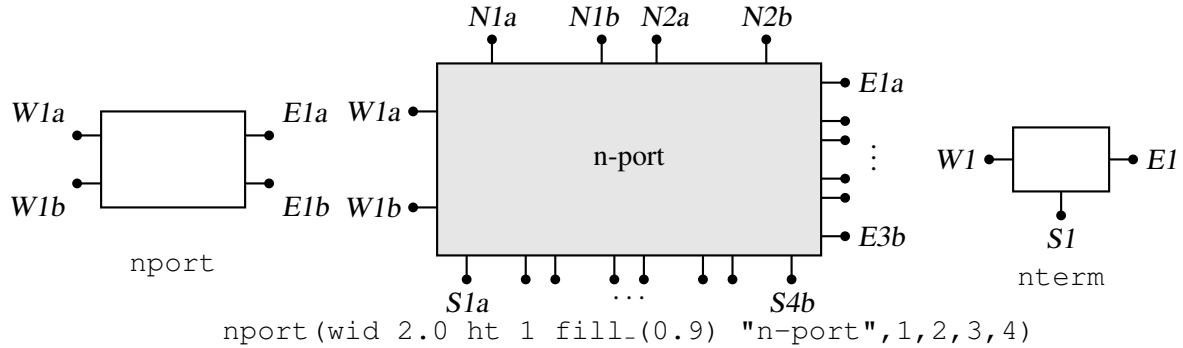
Figure 37: The nport and nterm macros [Nport.m4].
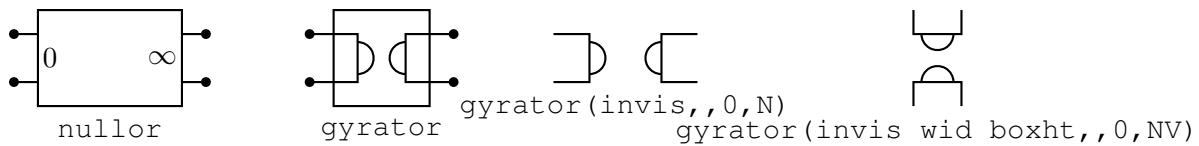
Figure 38: Some customizations of nport [NLG.m4].

11
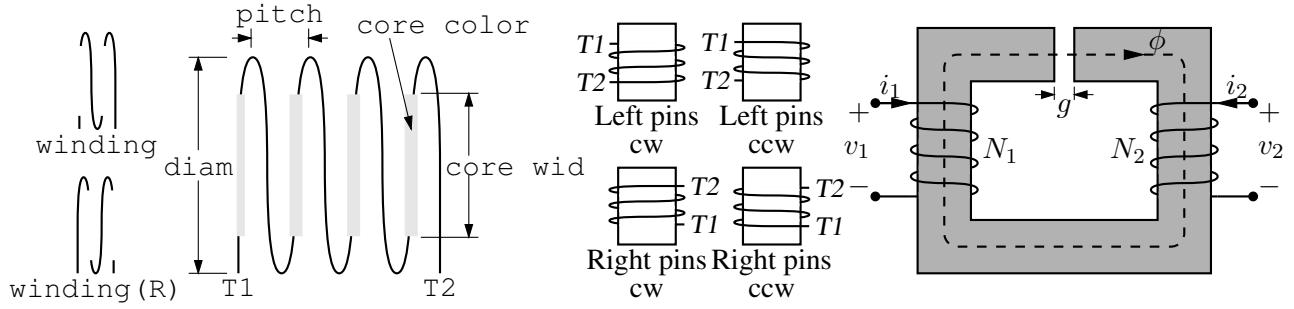
Figure 39: The macro `winding(L|R,diam,pitch,turns,core wid,core color)` [Windings.m4].
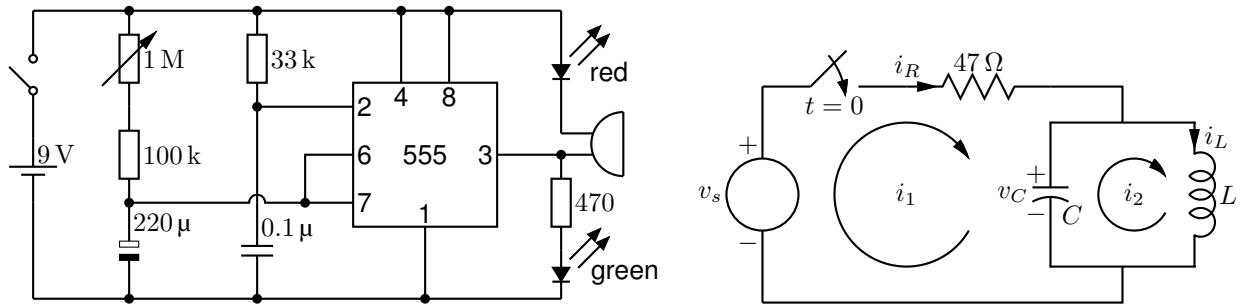

Figure 40: Two simple labeled circuits [ex01.m4].
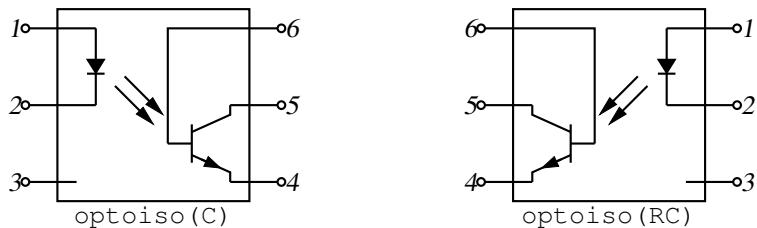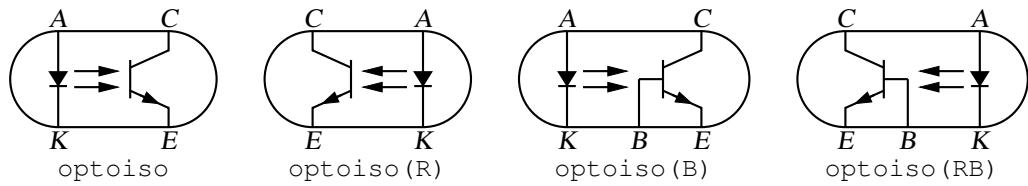
Figure 41: Optical isolator: a circuit with right or left orientation [Optoiso.m4].
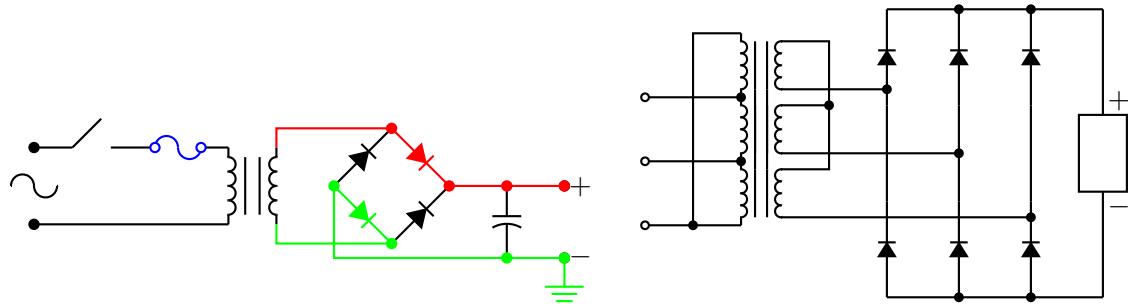
Figure 42: An elementary power supply circuit with colored elements, and a multiple-winding transformer with 3-phase rectifier [pwrsupply.m4].
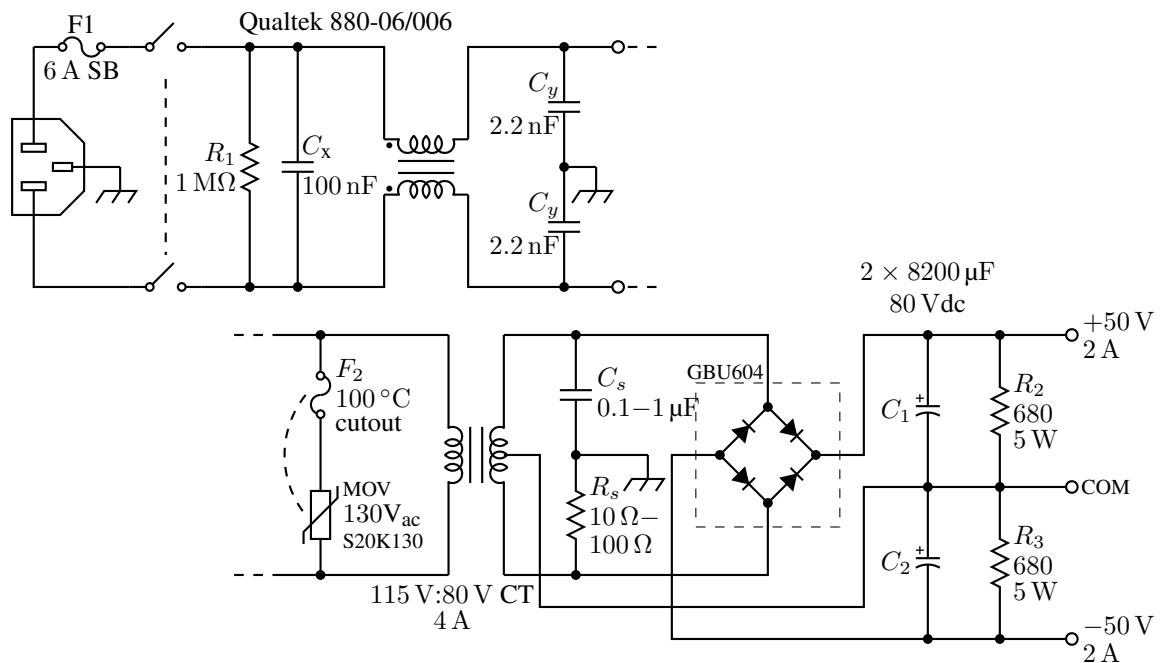
Figure 43: An unregulated 50V power supply [PS50.m4].

13
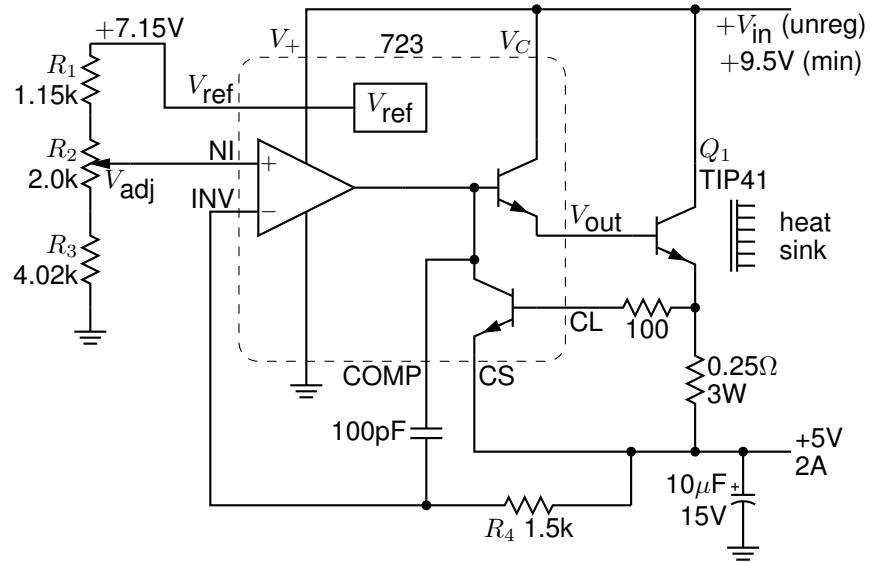
Figure 44: A 723 regulator circuit with a heat sink [Reg723.m4].
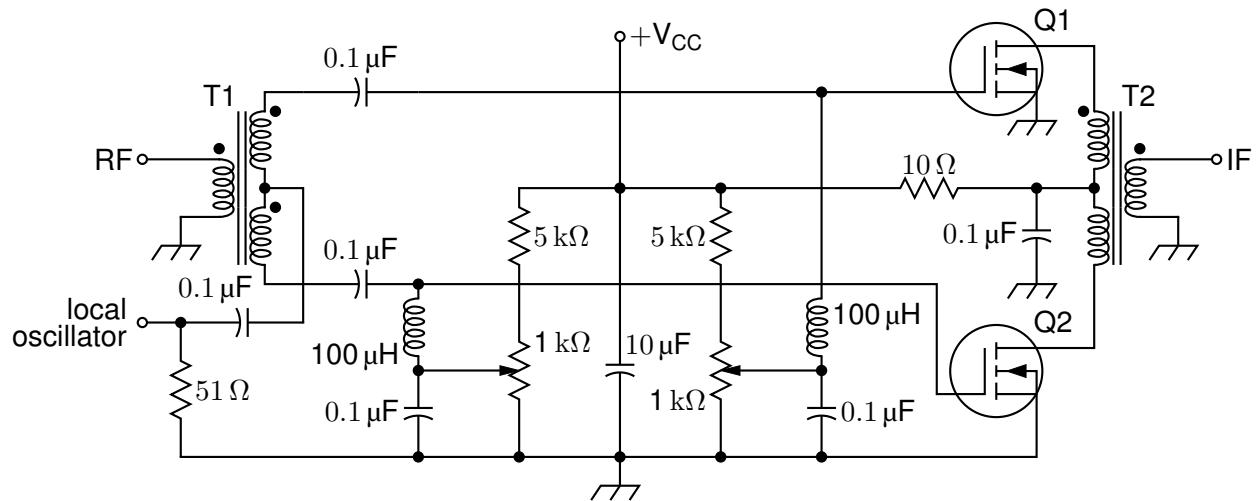
Figure 45: A balanced mixer, using `mosfet` and a custom transformer [Mixer.m4].

14
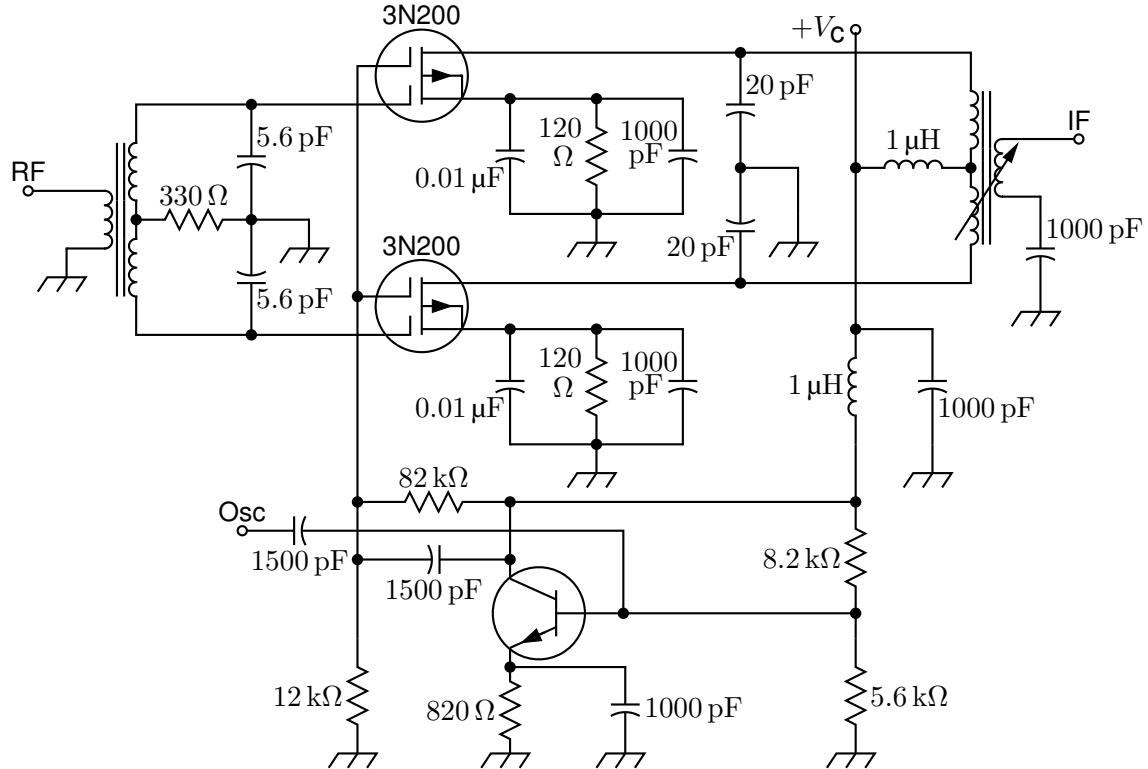
Figure 46: A push-pull mixer, showing FETs with multiple gates [PushPull.m4].
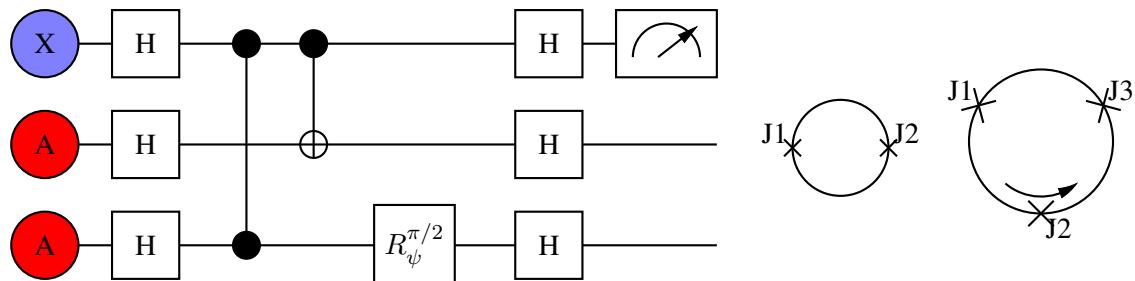
Figure 47: A quantum circuit and a superconducting quantum interface device (drawing direction down) [Quantum.m4].
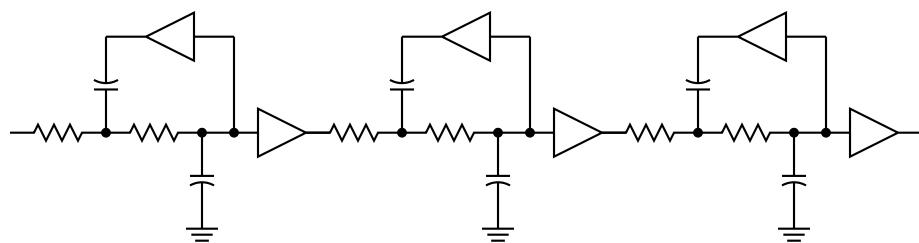
Figure 48: A six-pole filter [Sixpole.m4].

15
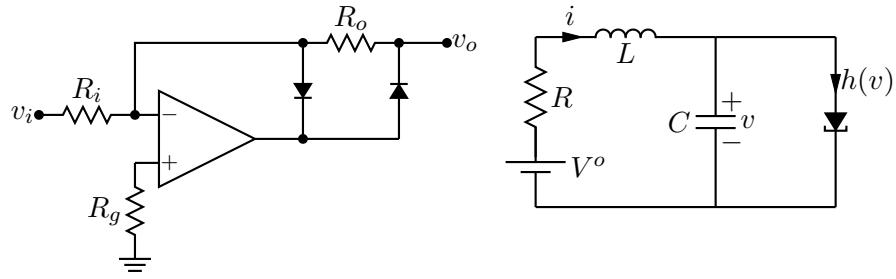
Figure 49: Precision half-wave rectifier and a tunnel diode circuit (illustrating `opamp, diode, resistor, ground,` and labels) [ex18.m4].
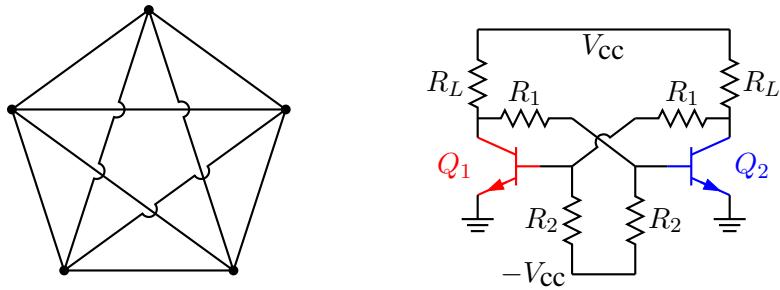
Figure 50: Non-planar graph and bistable circuit (illustrating the `crossover` macro and colored elements) [ex10.m4].
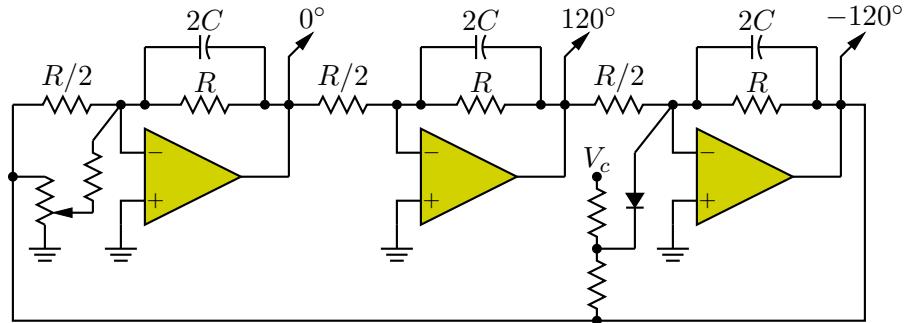
Figure 51: Three-phase oscillator [Three.m4].
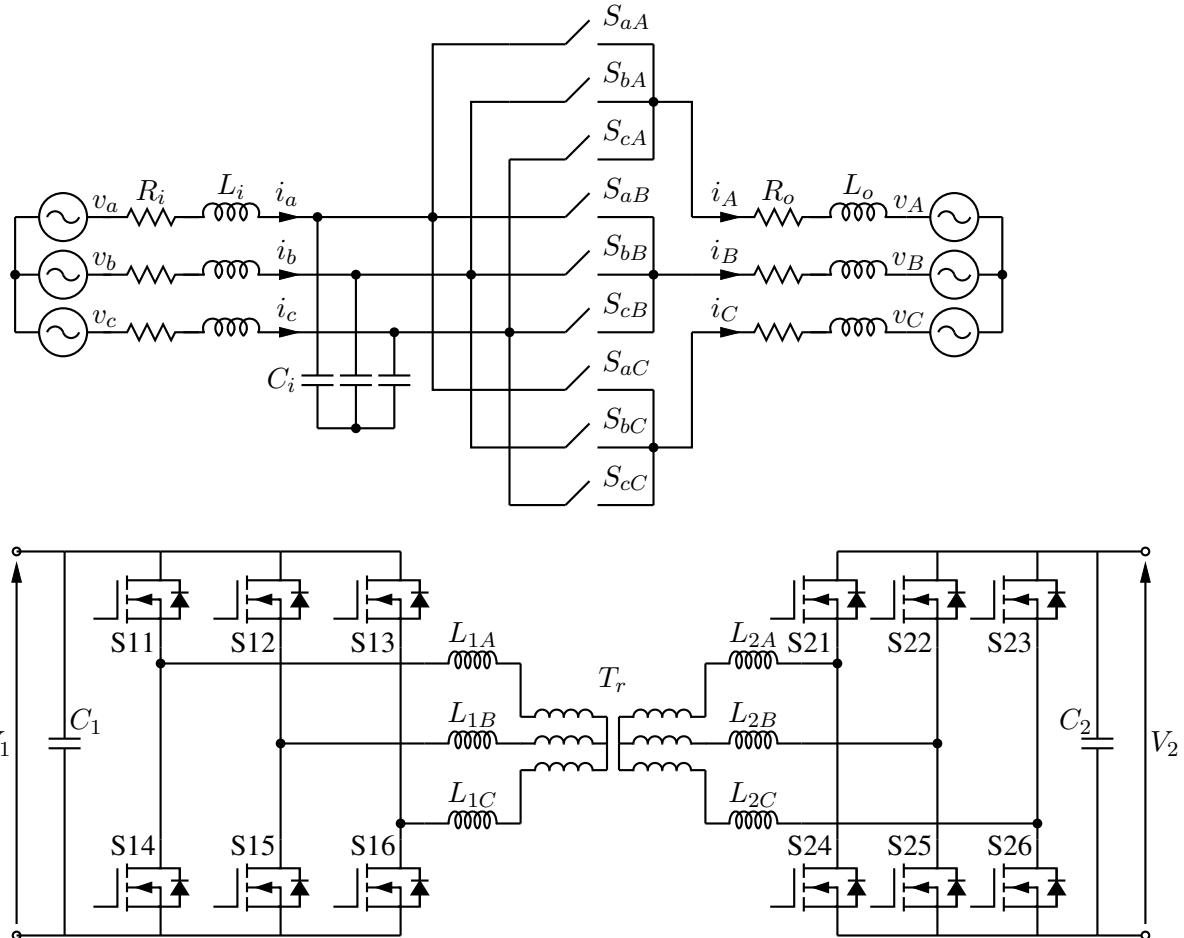
Figure 52: A three-phase switched AC-AC converter and a DC-DC converter [MC.m4].
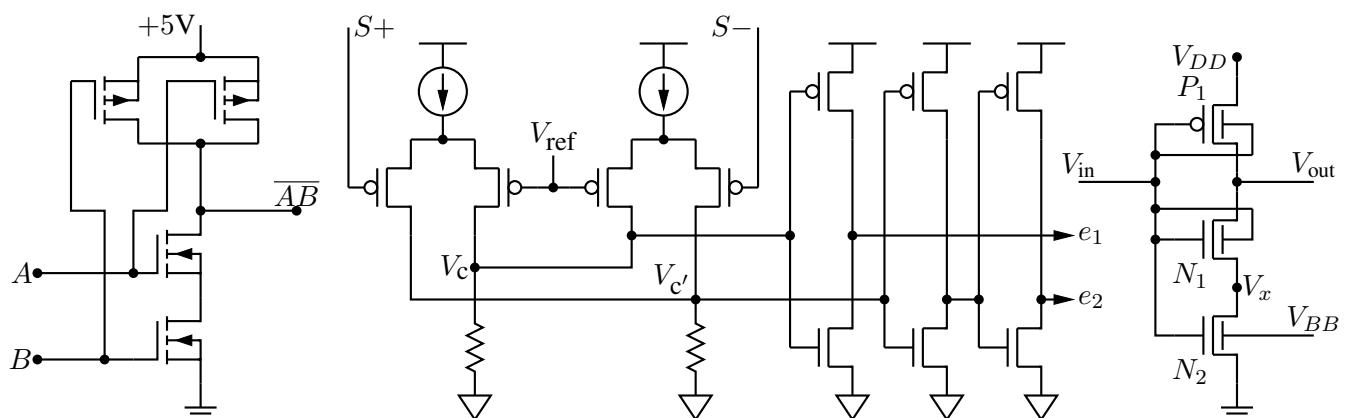
Figure 53: A CMOS NAND gate, a test circuit, and an XMOSFET example [ex12.m4].
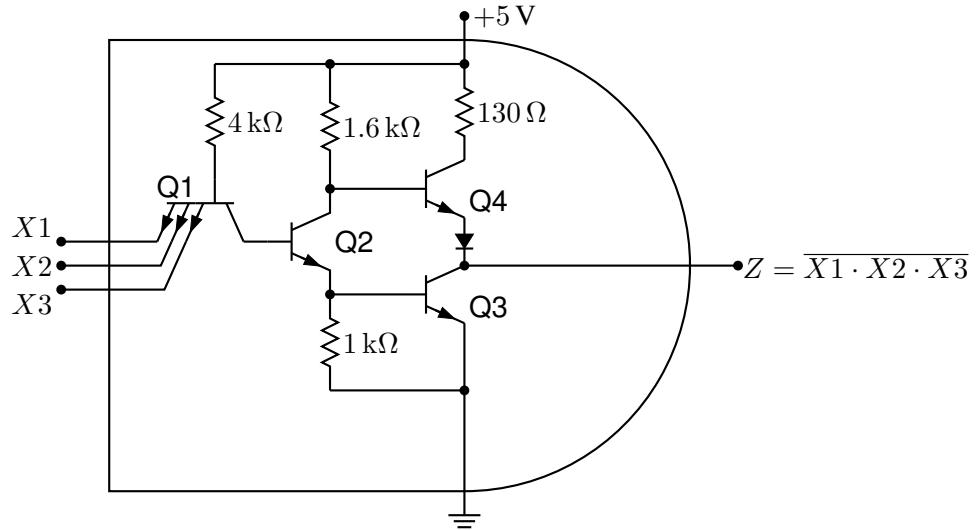
Figure 54: TTL NAND gate illustrating a transistor with multiple emitters [`TTLnand.m4`].
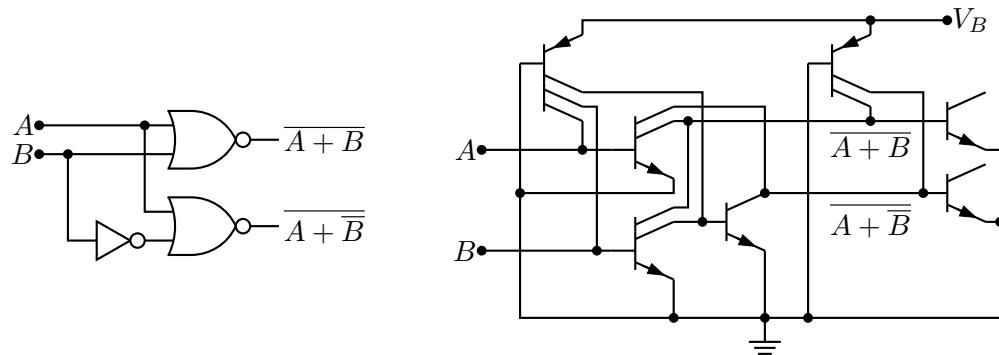
Figure 55: Gate circuit and equivalent embedded I^2L components illustrating multiple collectors [`I2L.m4`].
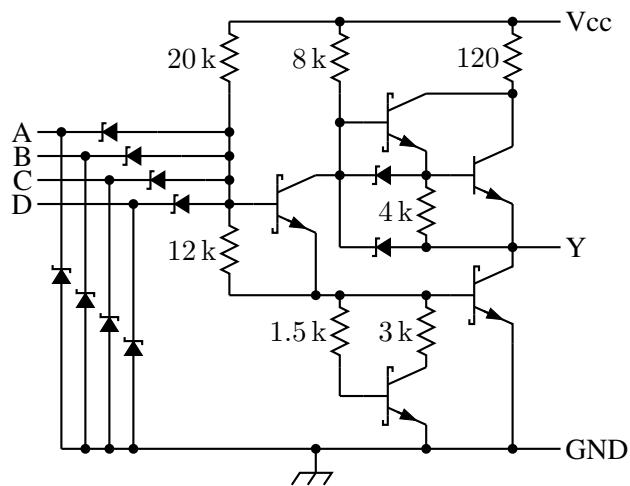

Figure 56: A 4-input NAND circuit illustrating the S (Schottky) option of bi_trans [`Schottky.m4`].

18
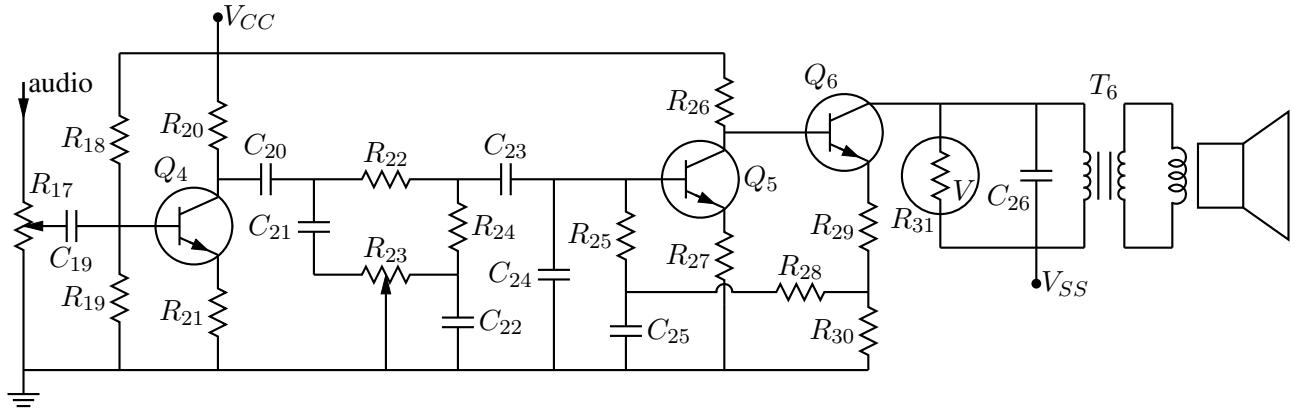
Figure 57: Transistor radio audio chain [ex11.m4].
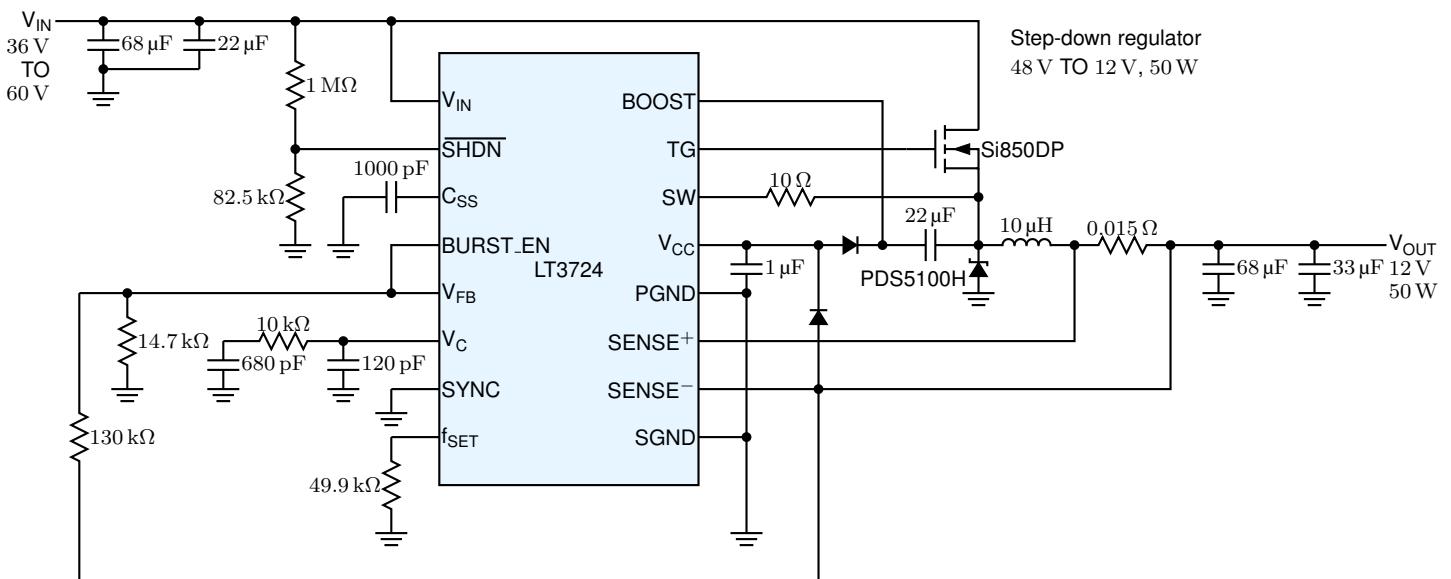
Figure 58: IC controller and auxiliary elements of a step-down converter [LT3524.m4].
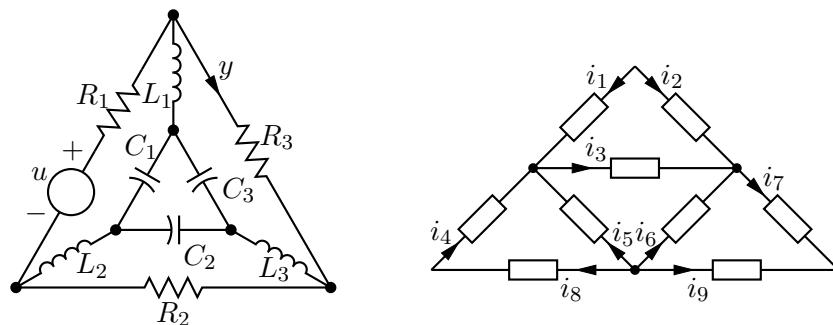
Figure 59: Labels on non-manhattan elements [ex04.m4].
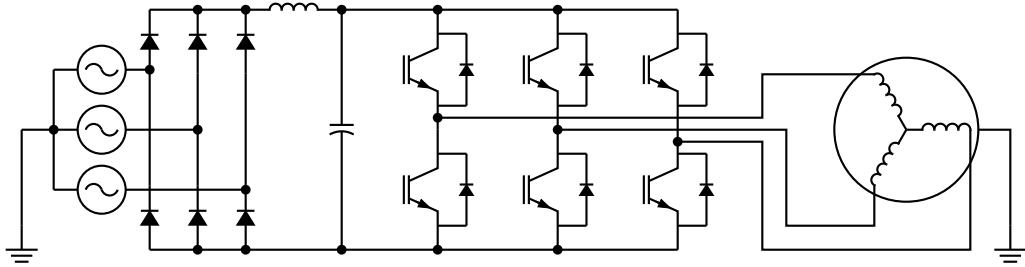
Figure 60: Synchronous machine driven by variable-speed drive and rectifier [Drive.m4].
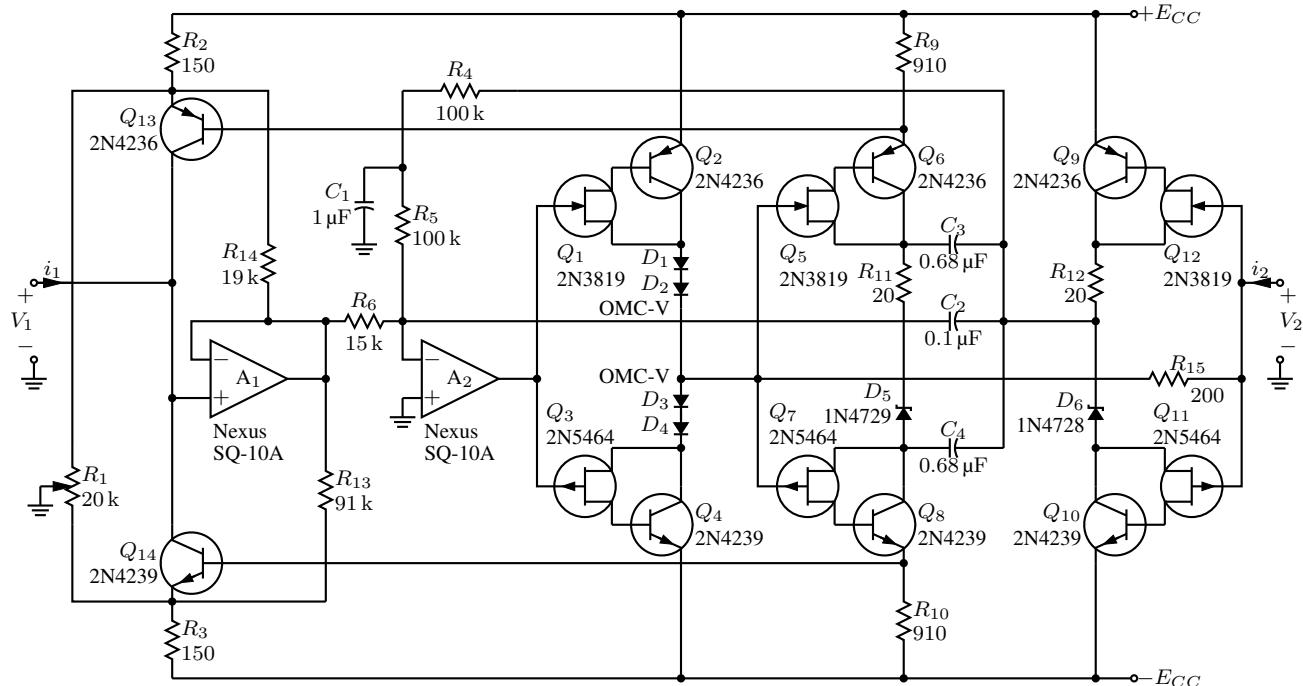

Figure 61: Realization of a controlled source (illustrating stacked element labels) [Csource.m4].
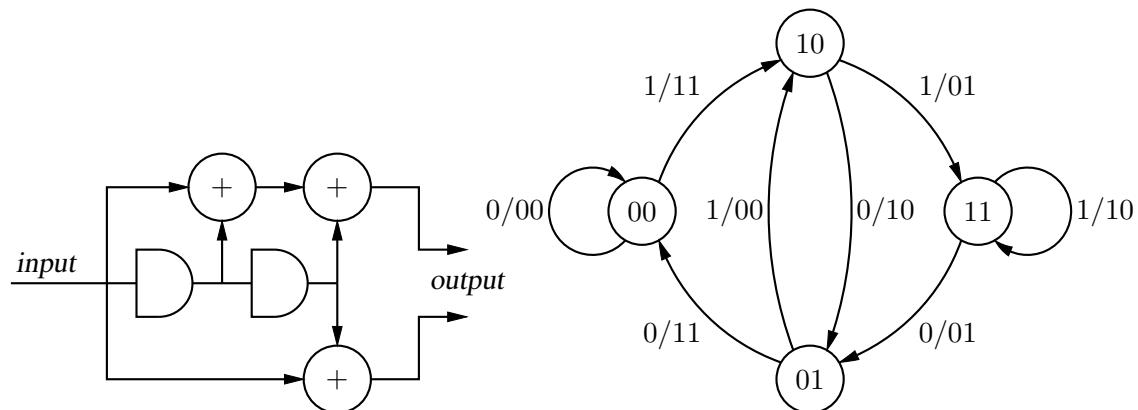

Figure 62: A rate $1/2$ binary convolutional coder and its state diagram [ex16.m4].
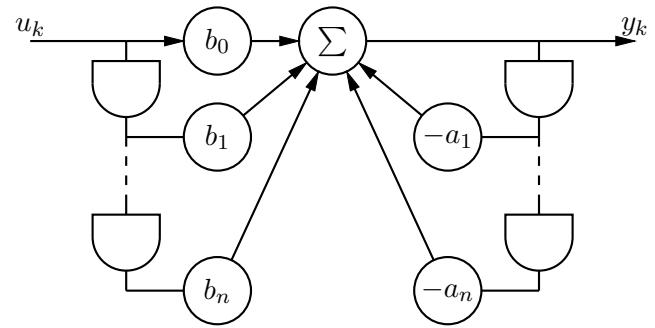
Figure 63: Digital filter [ex03.m4].
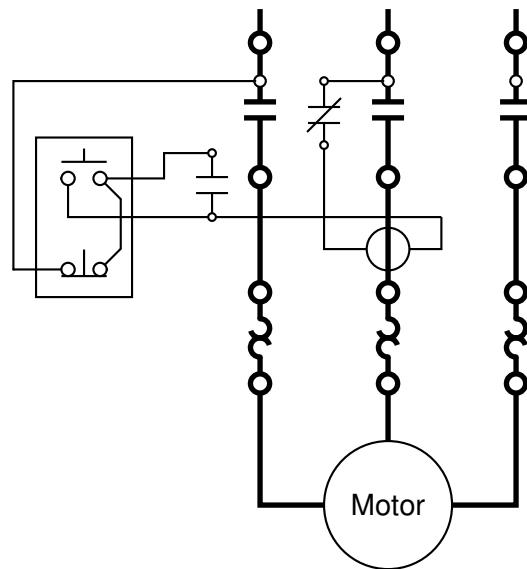
Figure 64: Motor control connections [MotorControl.m4].
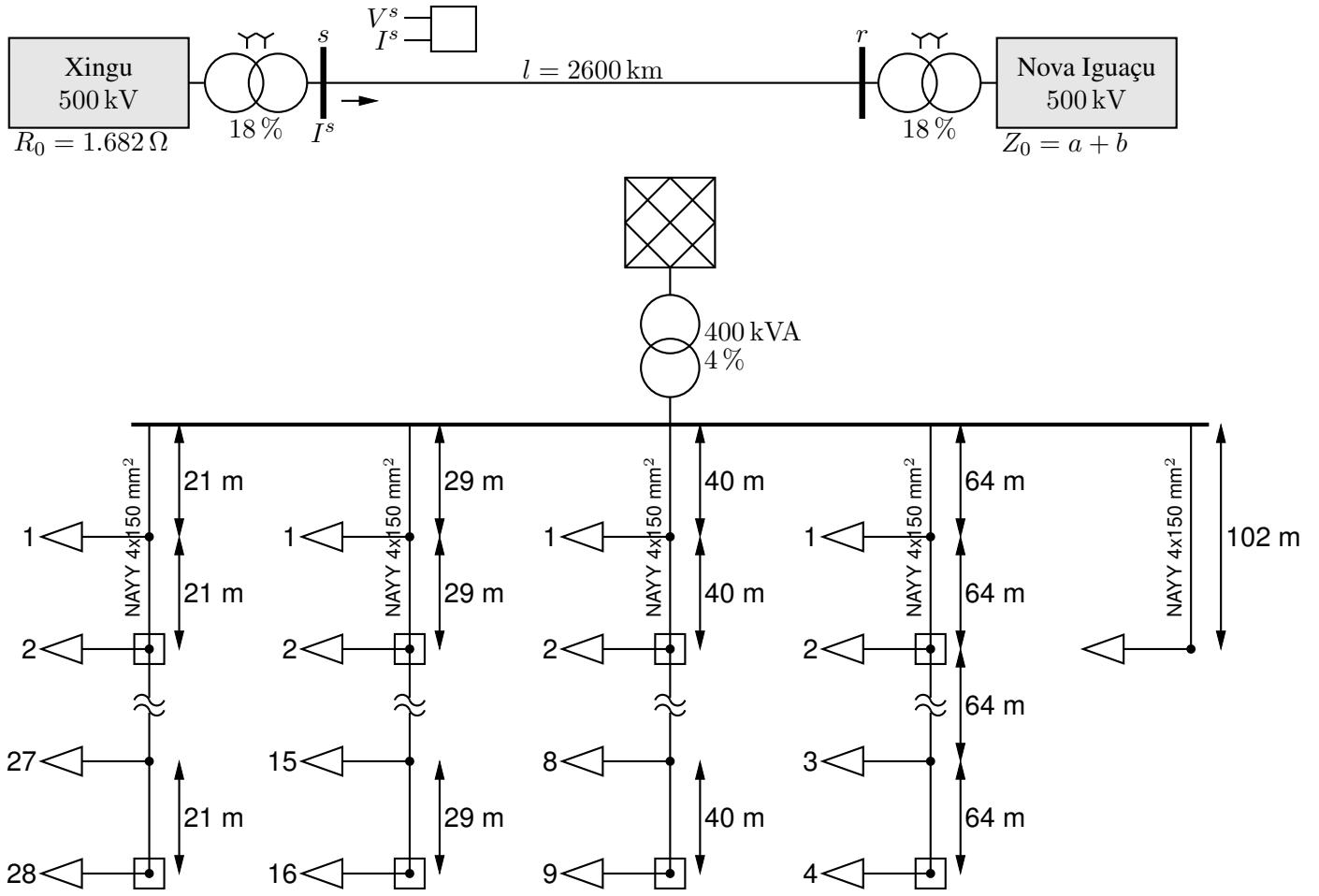
Figure 65: Single line distribution diagram [`OneLine.m4`].
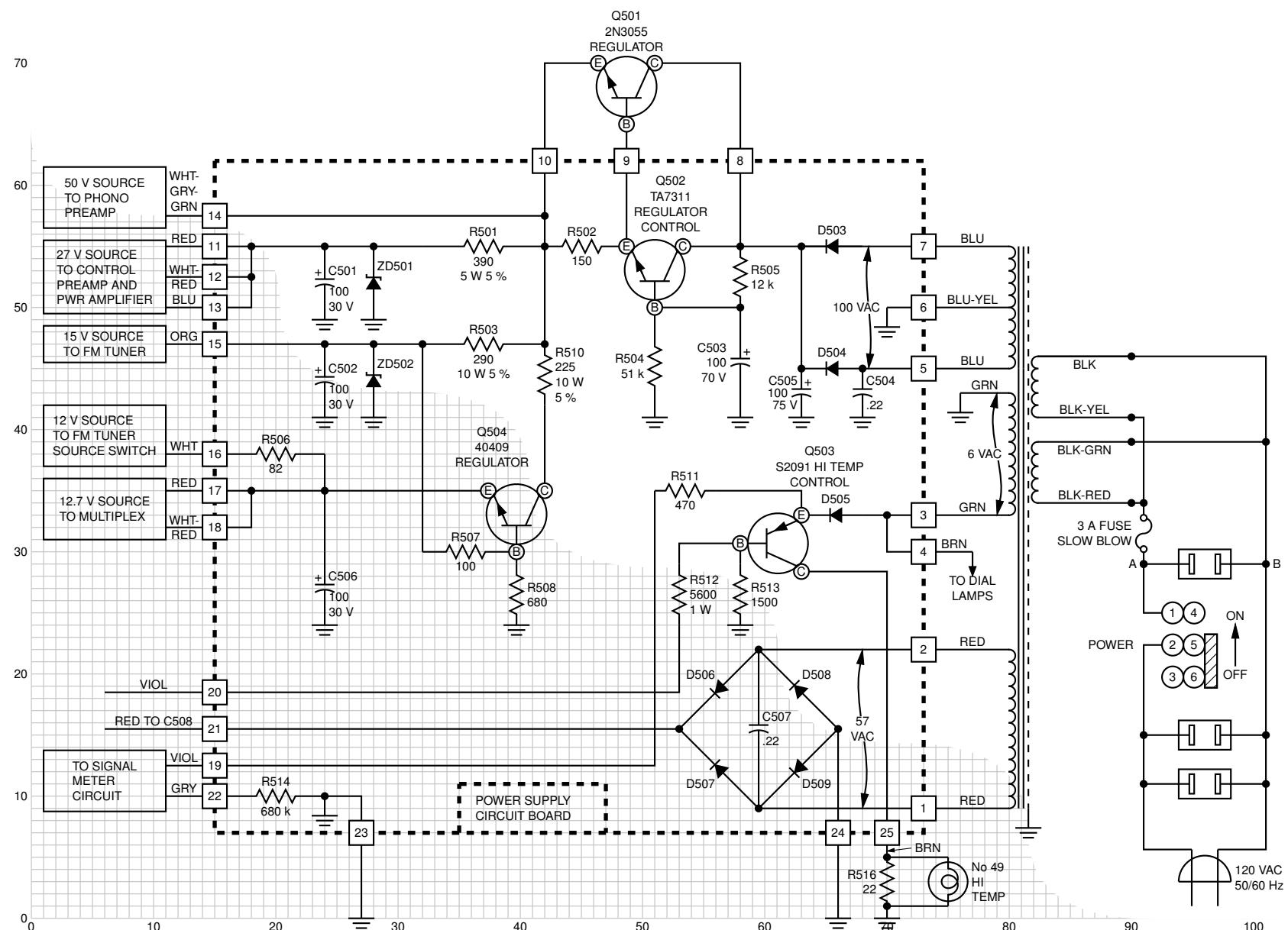
Figure 66: The power supply of a Heathkit AR-15 (Now, *that* was a receiver!) with custom transformer and other elements, drawn on a grid (partially shown) to aid in placement [Heathkit.m4].
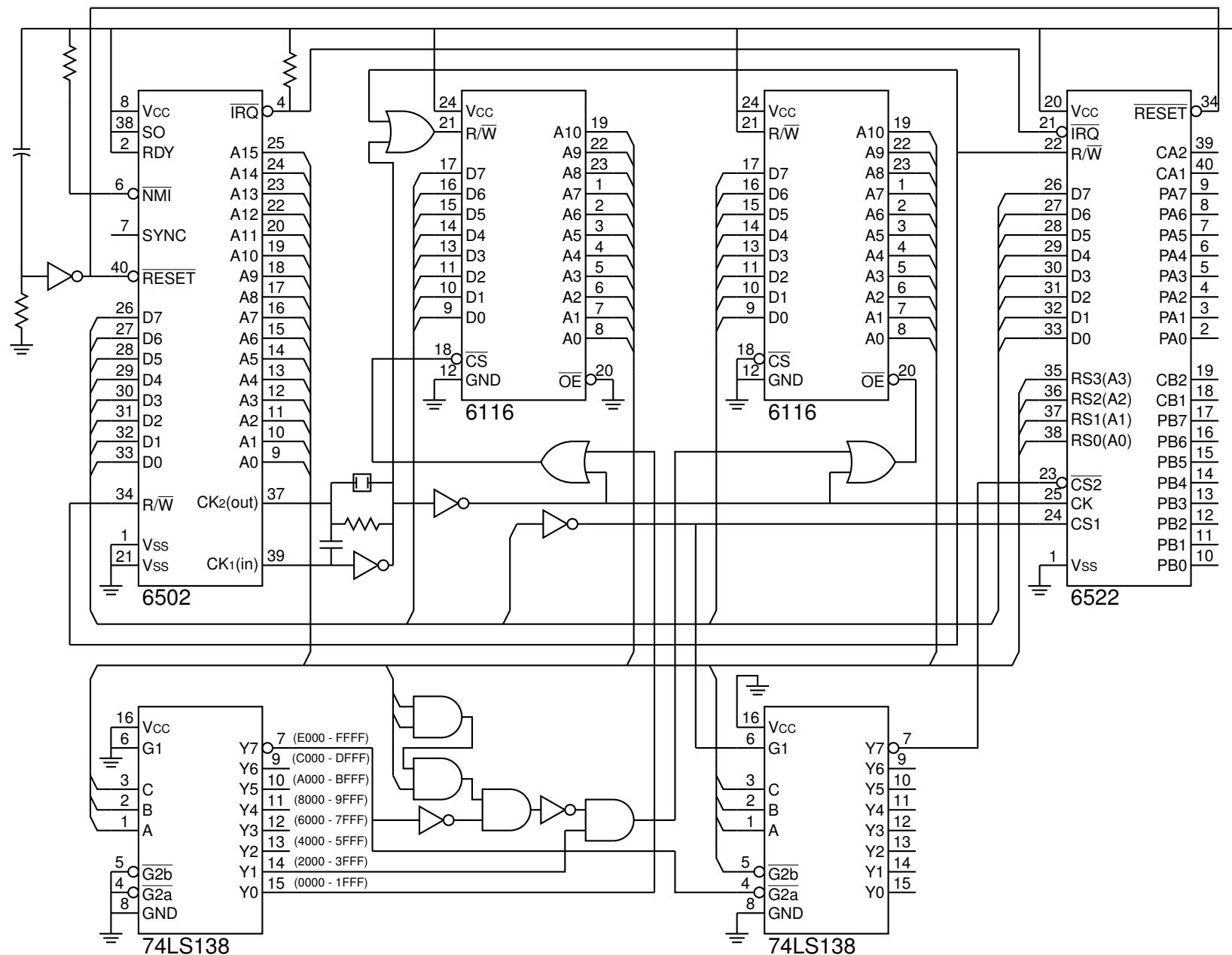
Figure 67: A digital circuit of moderate size, redrawn from M. P. Maclenan and G. M. Burns, "An Approach to Drawing Circuit Diagrams for Text Books," Tugboat (12)1, March 1991, pp. 66-69 [lcct.m4].
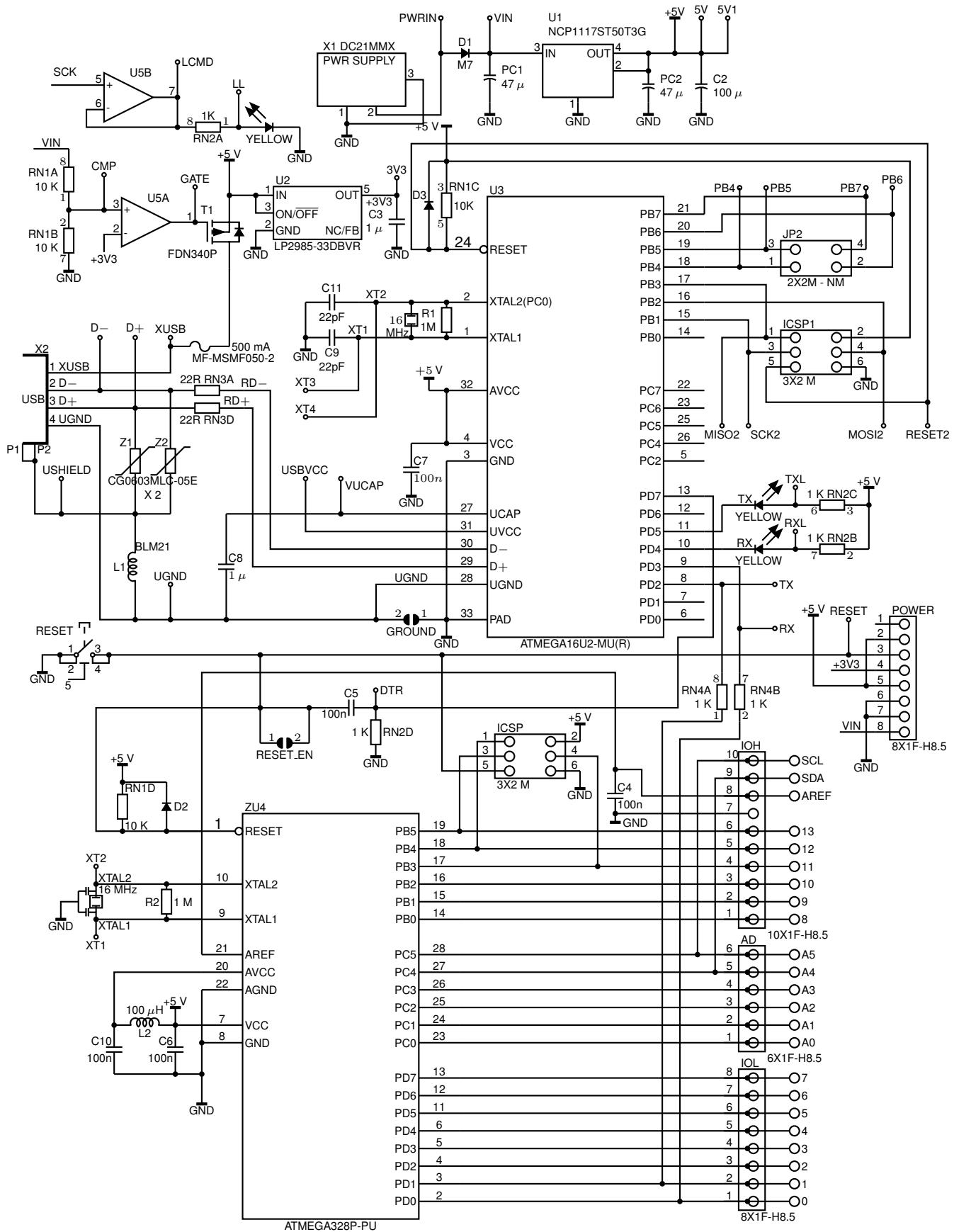
Figure 68: An Arduino UNO circuit adapted and redrawn [`UNO.m4`].
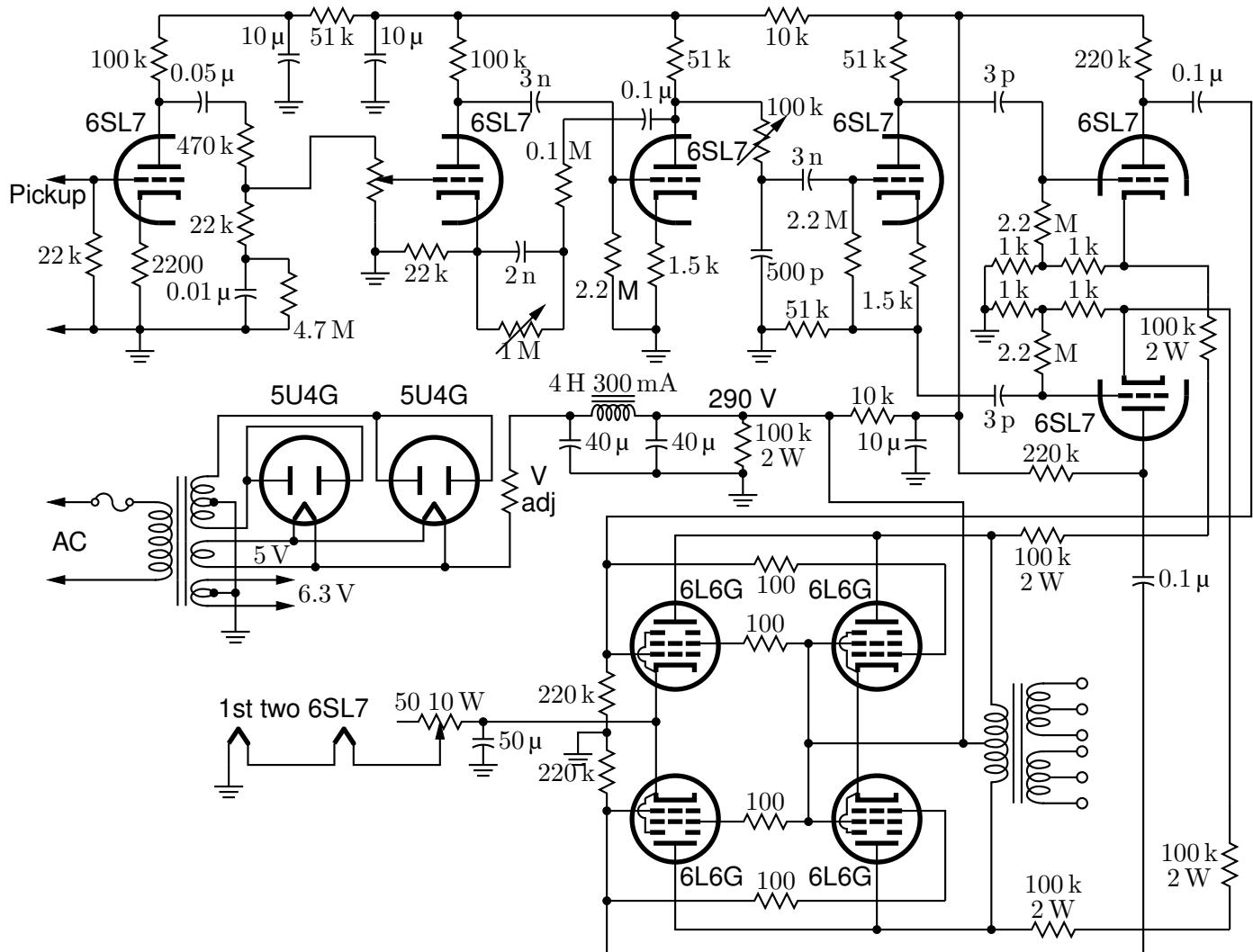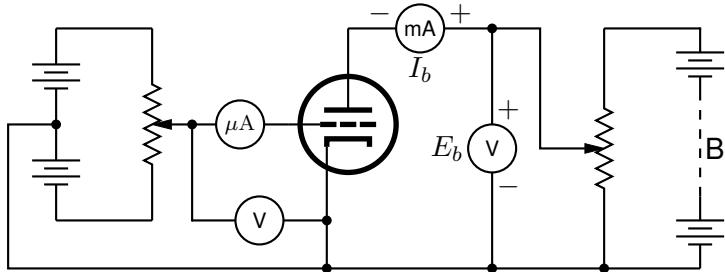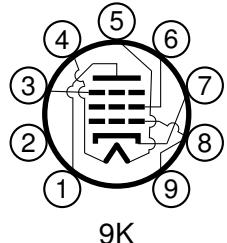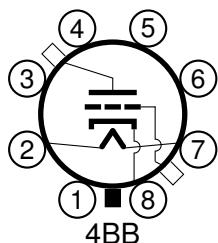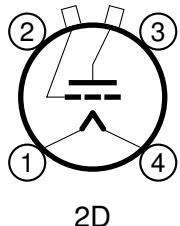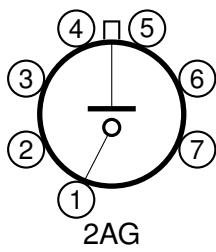
Figure 69: Electron-tube diagrams: a few bottom-view base diagrams, a generic triode test circuit, and a 25-watt audio amplifier adapted from F. Langford-Smith, *Radiotron Designer's Handbook,* fourth edition, Harrison, NJ: Radio Corporation of America, 1952 [Tubediags.m4].
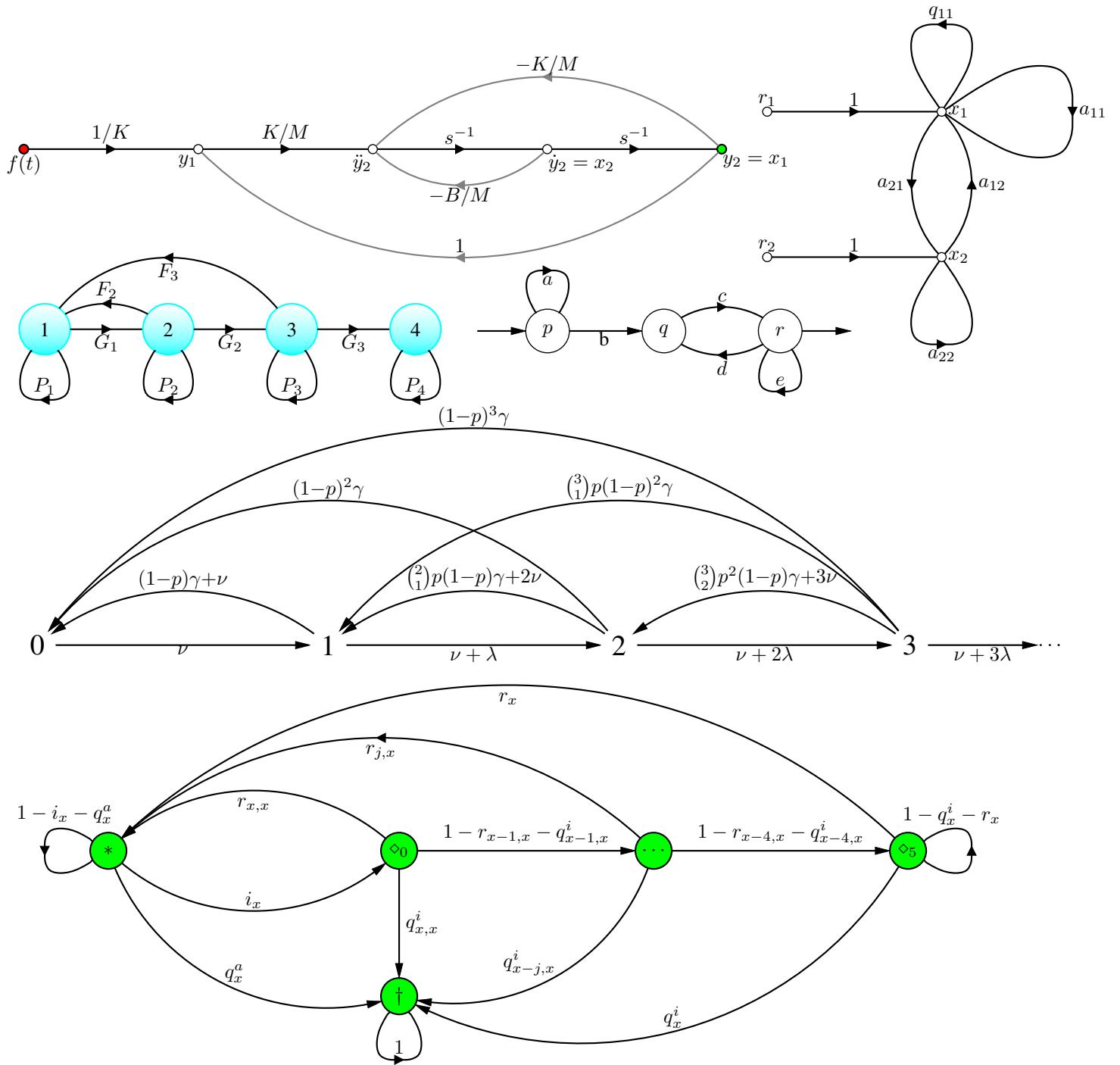
Figure 70: Signal-flow graphs [`sfg.m4`].
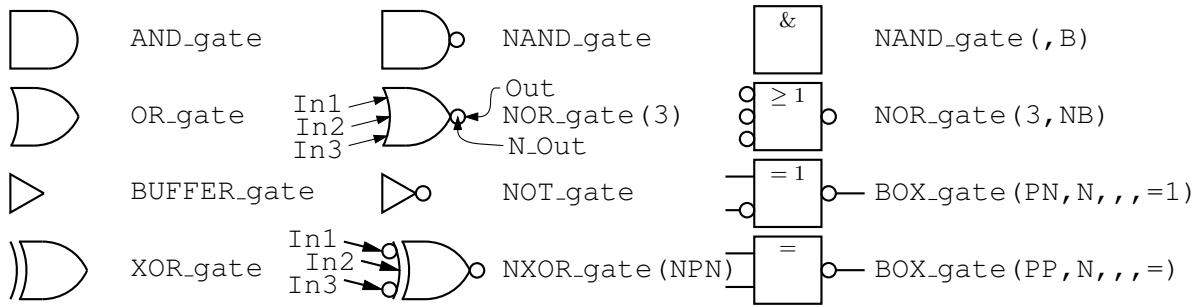
Figure 71: Basic logic gates [Logic.m4].


To other latches
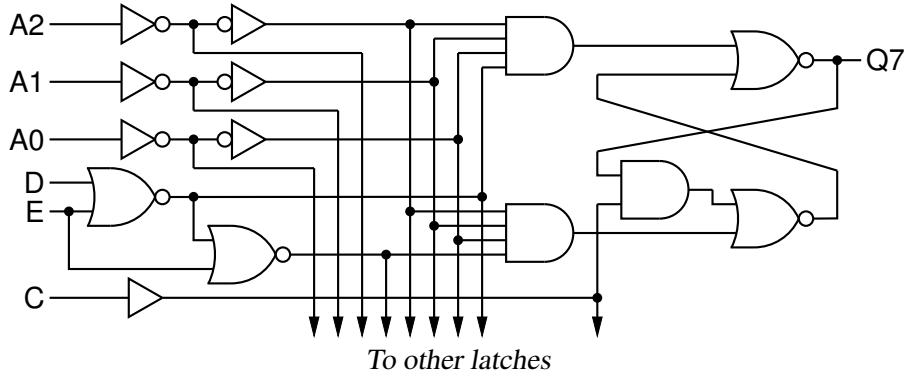
Figure 72: General-purpose latch: a small logic circuit [ex08.m4].
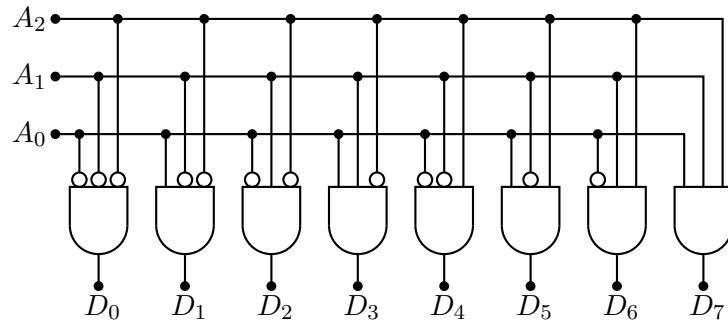

Figure 73: Decoder logic, constructed using the for_ macro [Decoder.m4].
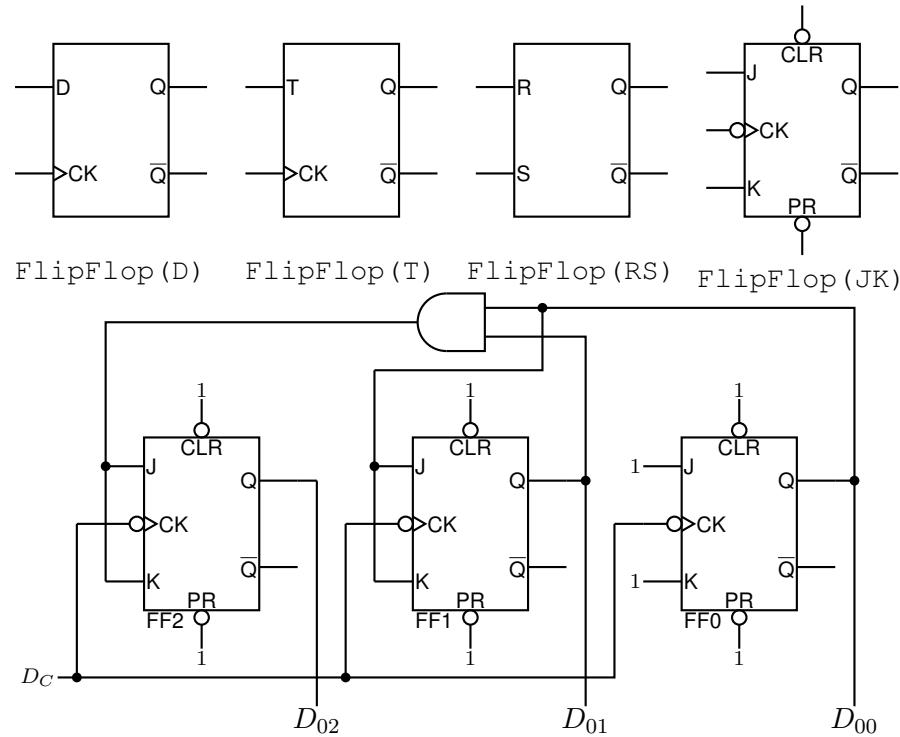
Figure 74: Some flip-flops [ex21.m4].
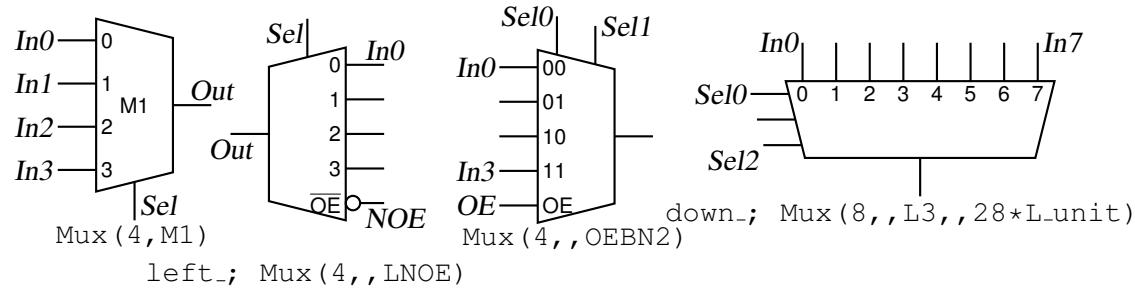
Figure 75: Multiplexer [Multiplexer.m4].
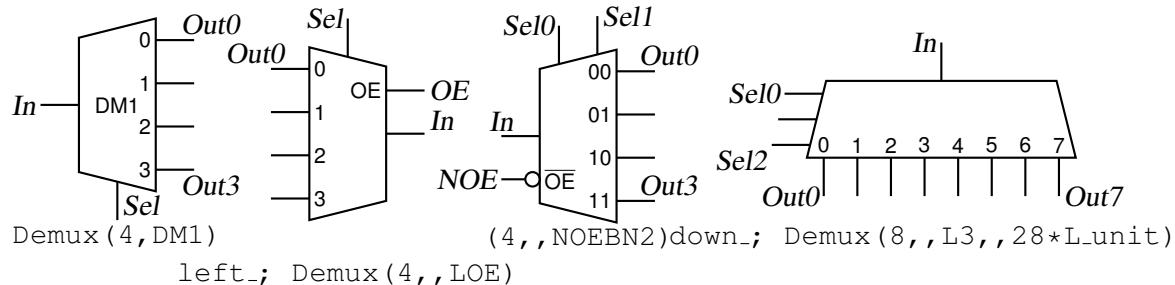

Figure 76: Demultiplexer [Demultiplexer.m4].
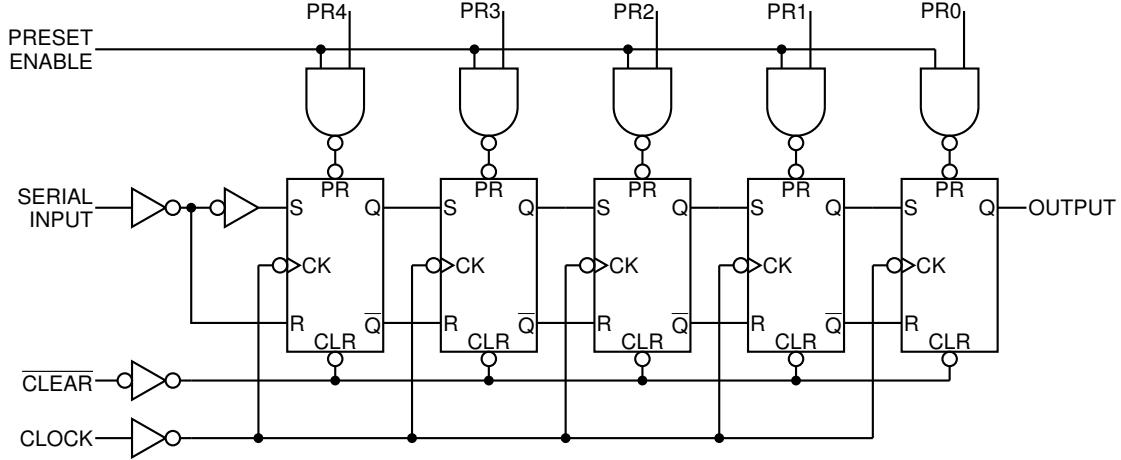
Figure 77: A 5-bit shift register drawn using a custom flip-flop [ShiftR.m4].

| 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
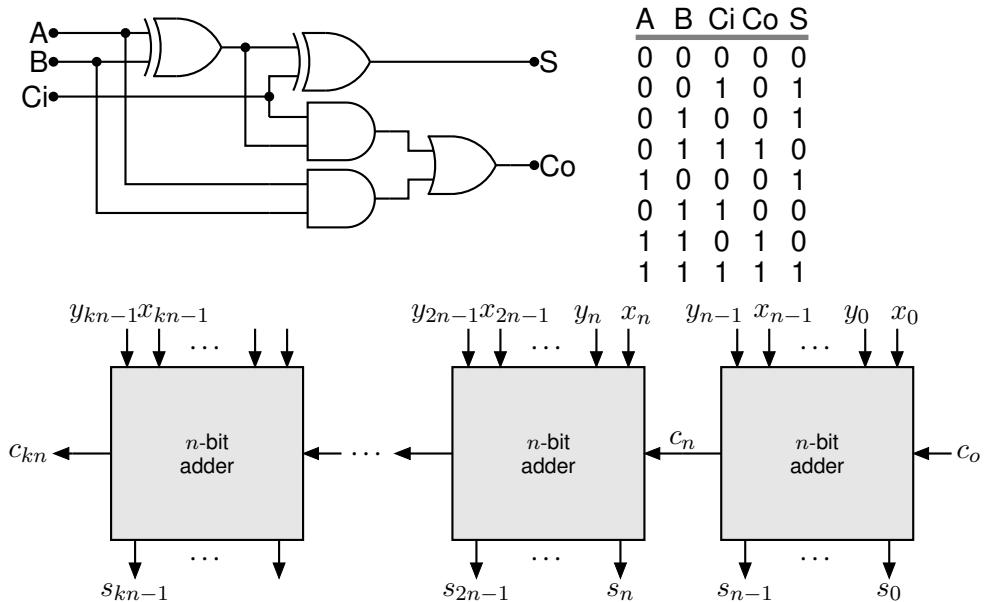
Figure 78: A full adder and a cascade of n-bit adders [Adder.m4].
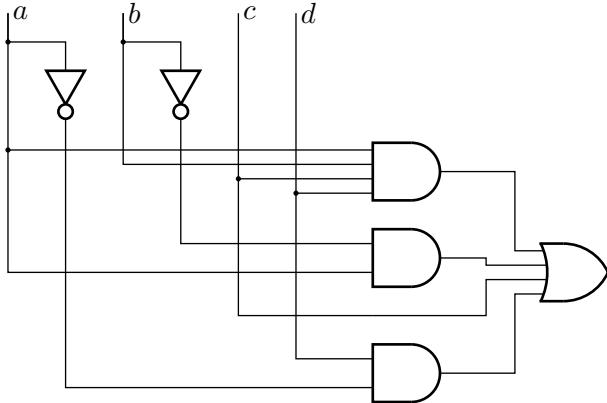
CanLogic(AND,,OR,,abcd,˜ba,c,d˜a)   CanLogic(OR,N,NAND,,ab˜c,a˜bc,ac,˜d)
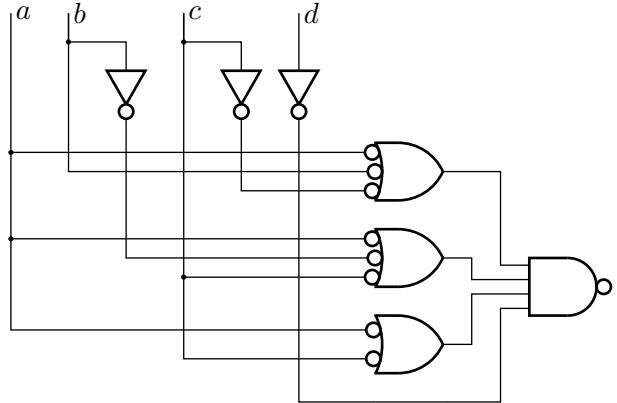
Figure 79: A way of automatically drawing two-layer logic diagrams [CanLogic.m4].

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

Autologix(
  Nand(And(Or(A,C),Or(B,˜C),Or(˜A,B,C)),Ck);
  Nand(˜A,B))@with .n at last [].s+(0,-2bp__))
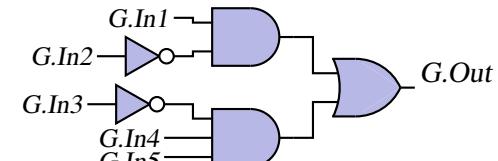
G: Autologix(
  Or(And(B,Not(C)),And(Not(A),B,C)),N,
 shaded "lightblue")

Figure 80: 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].

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

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

...,R)

...,LRV;offset=17.25*L_unit)
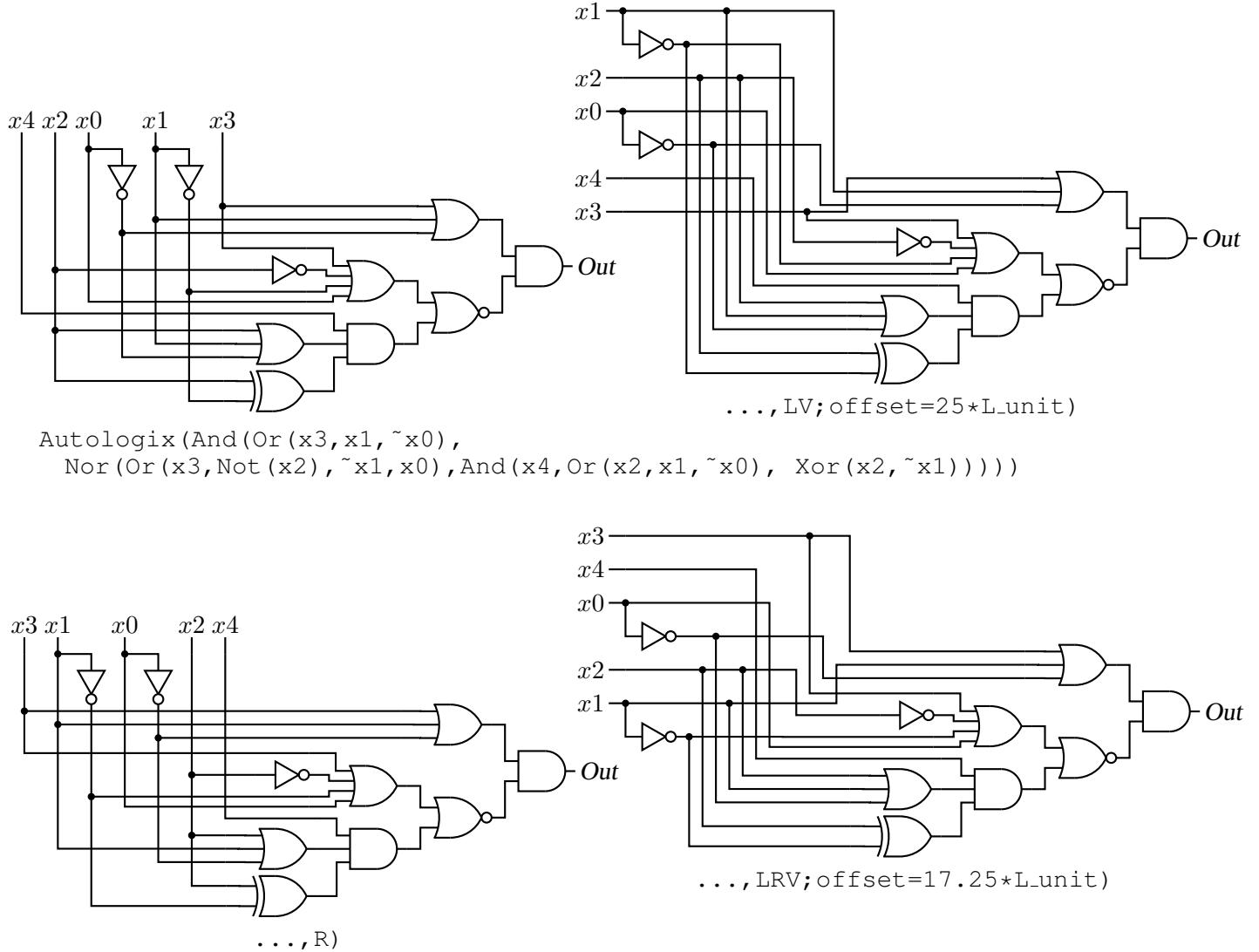
Figure 81: The Autologix macro can draw inputs on the left but the added drawing complexity may require hand tuning with second-argument options: L puts the inputs on the left, R reverses their order, V scans the input arguments in reverse order, and offset=*value* displaces the array of inputs [ABlogix.m4].
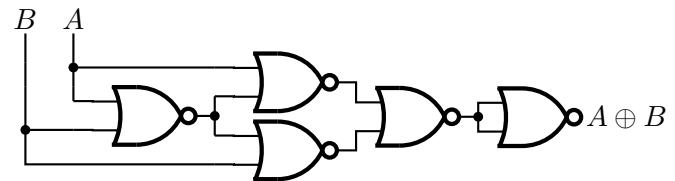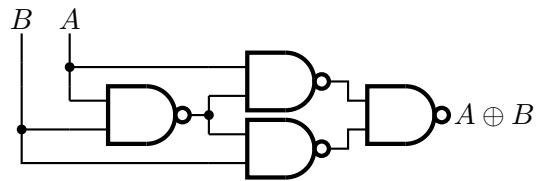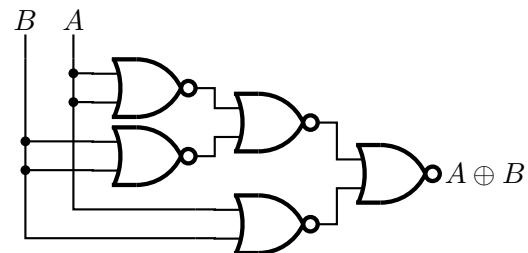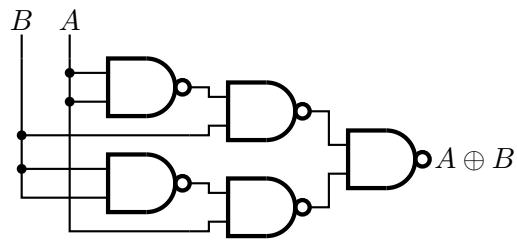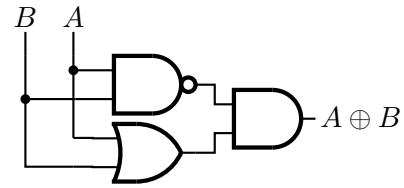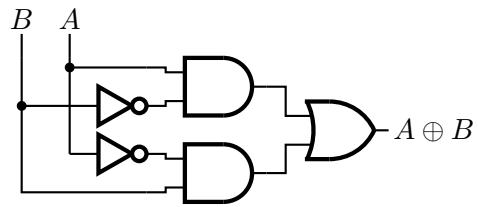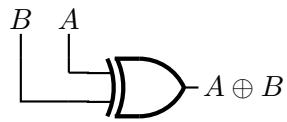
Figure 82: Realizations of the XOR function using `Autologix` [XOR.m4].
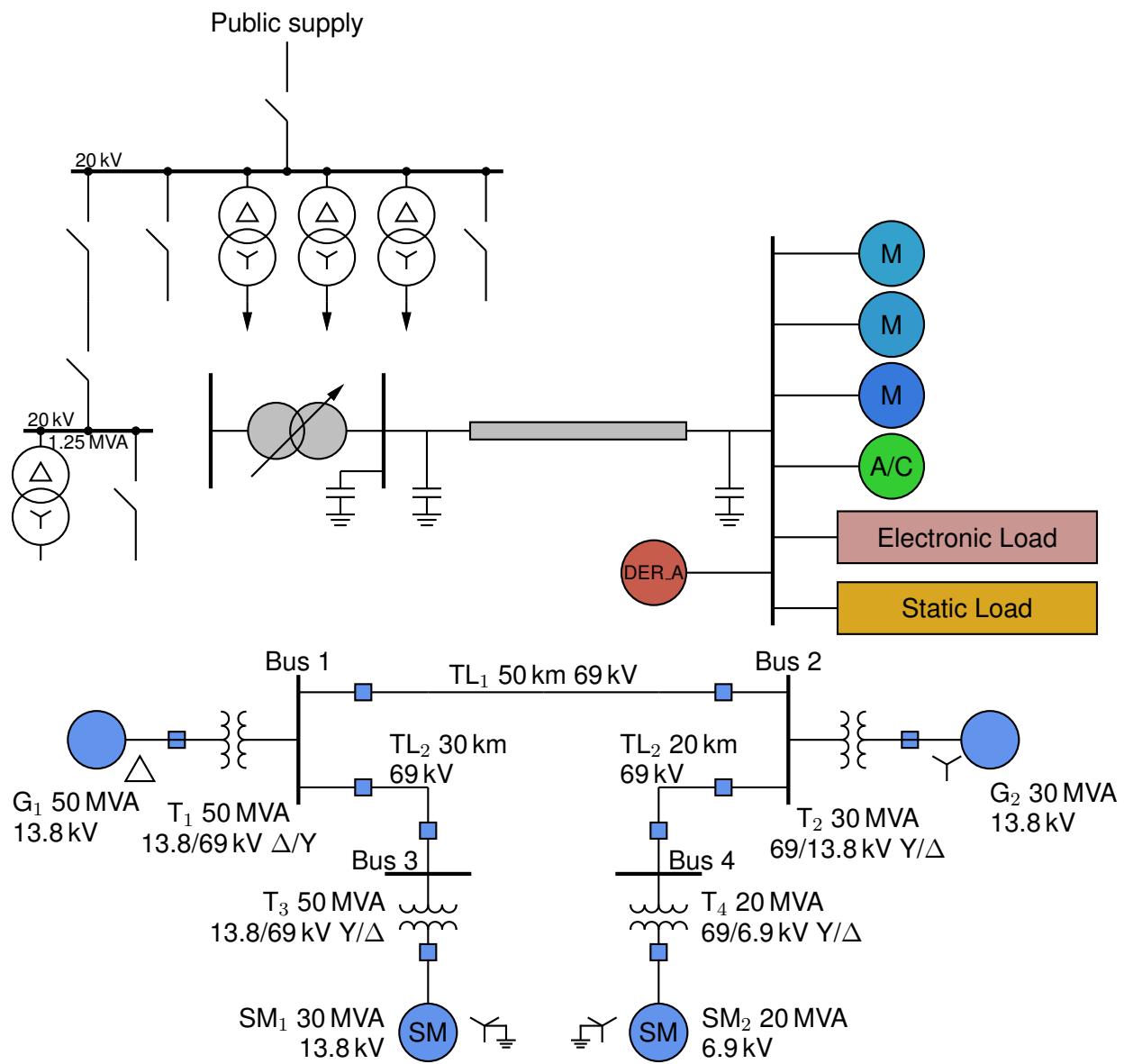
Figure 83: A test of single-line diagram macros [EEP.m4].
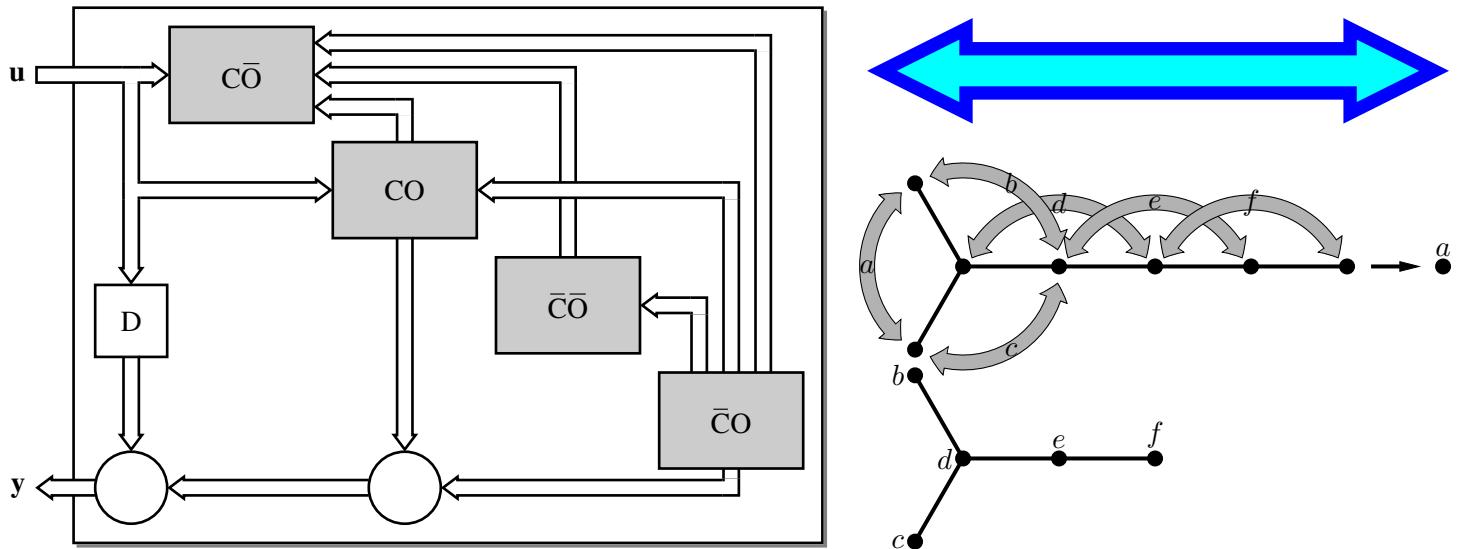
Figure 84: Use of `darrow` and `Darc` [`ex05.m4`].
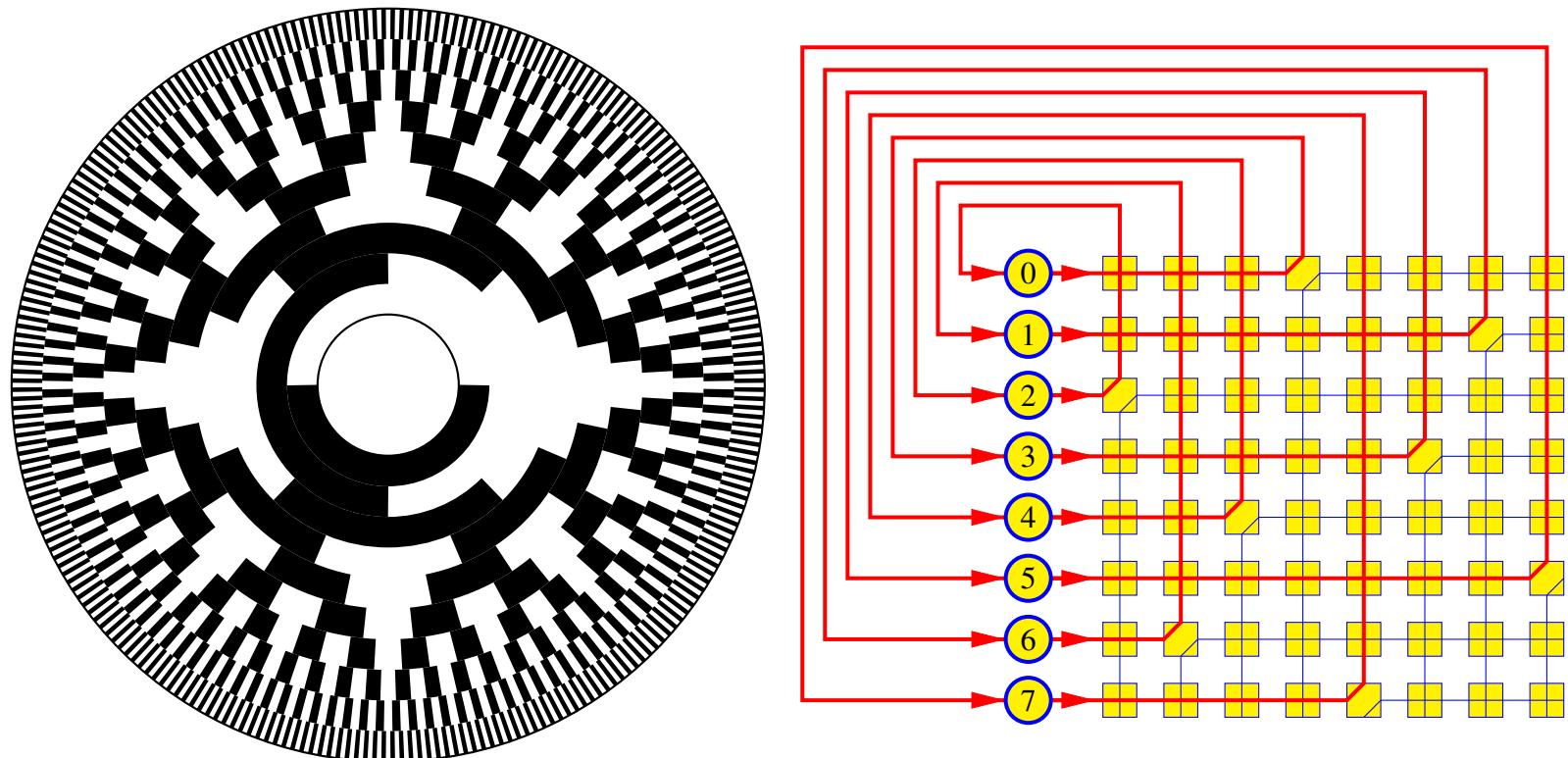
Figure 85: The power of looping and branching: Gray code 10-bit encoder disk pattern and a crossbar switch [`GrayCode.m4`].
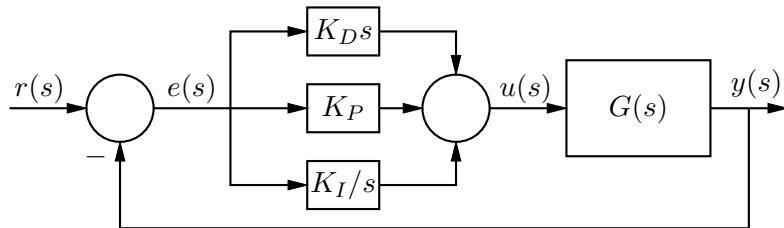
(a) PID control
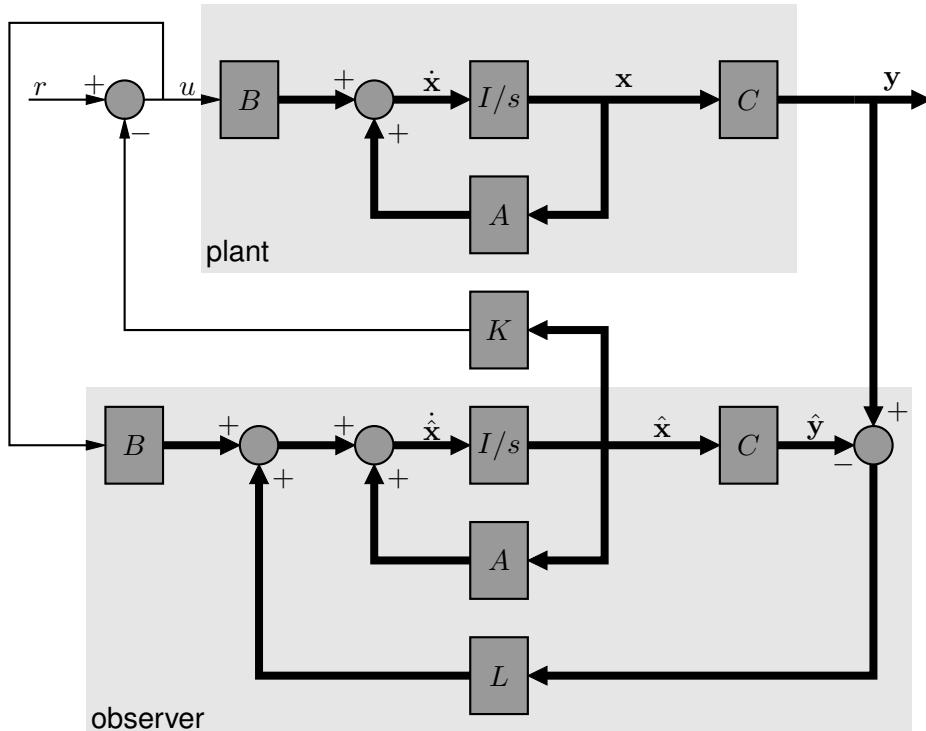
(b) Single-input plant with feedback from a full-order observer
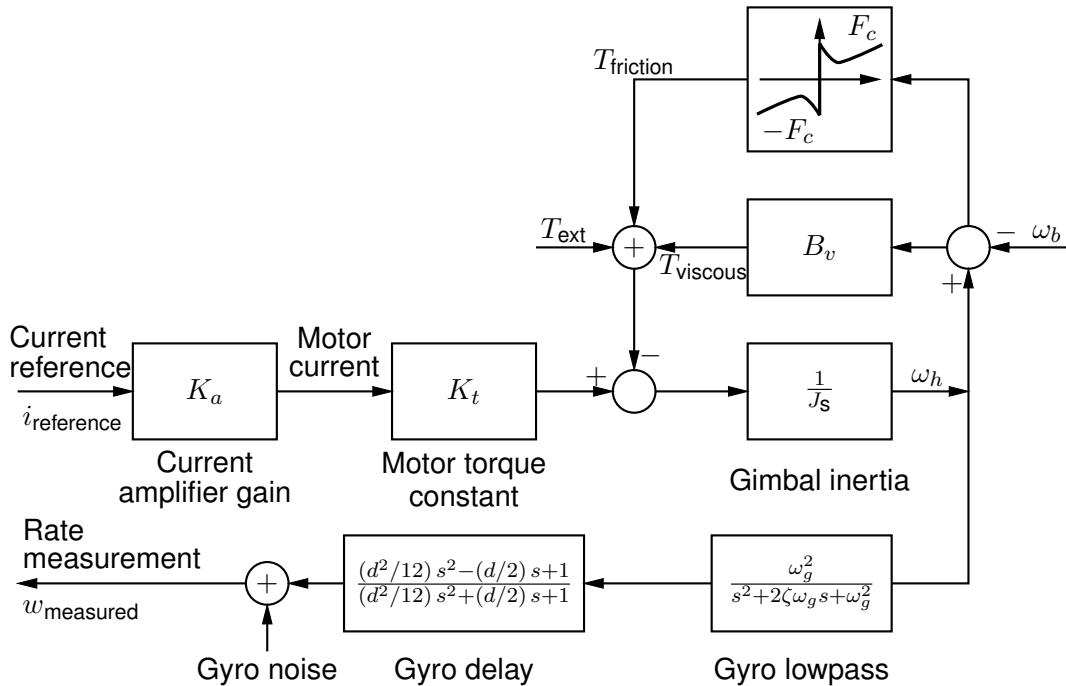
(d) Single-axis gimbal model

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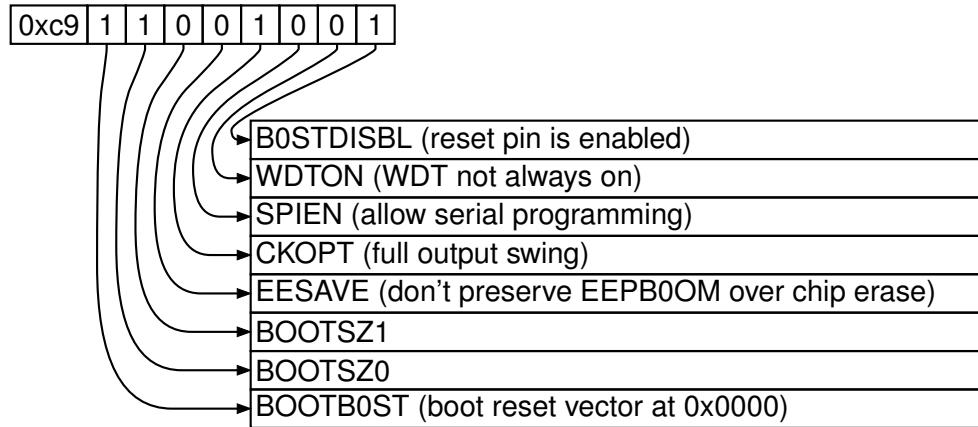
Figure 87: Elementary splines [Byte.m4].

Figure 88: A customizable seven-segment display showing the numbered segments, a custom shape, and the numerals from 0 to 9 [Sevensegment.m4].
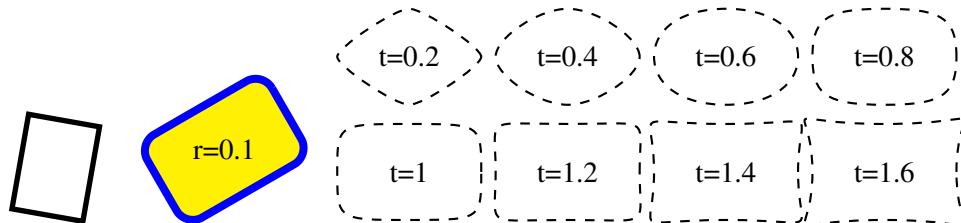
Figure 89: The macro rotbox(*wid,ht,type*,[r|t=*val*]) draws a box in the current direction [Rotbox.m4].
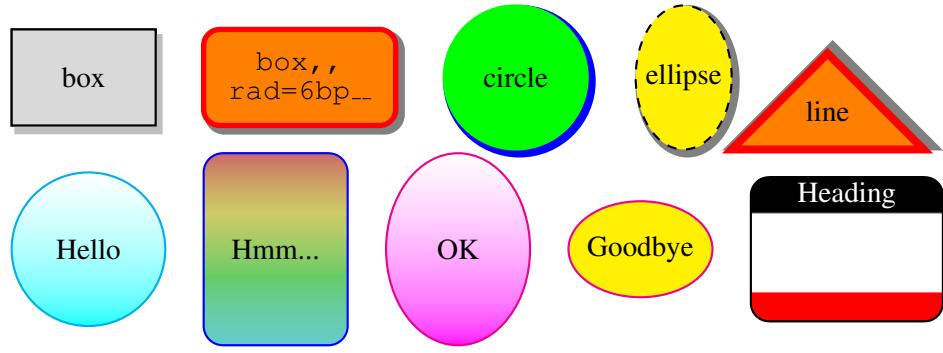
Figure 90: Embellishments: the top row illustrates `shadowed(box|circle|ellipse|line,[at po-sition],keys)` and the second row `ColoredV(box|circle|ellipse,(r,g,b)|((colorseq)),attributes)` `[Shadowed.m4]`.
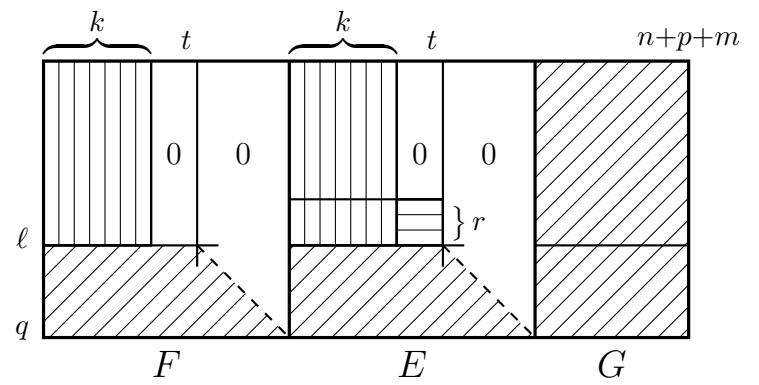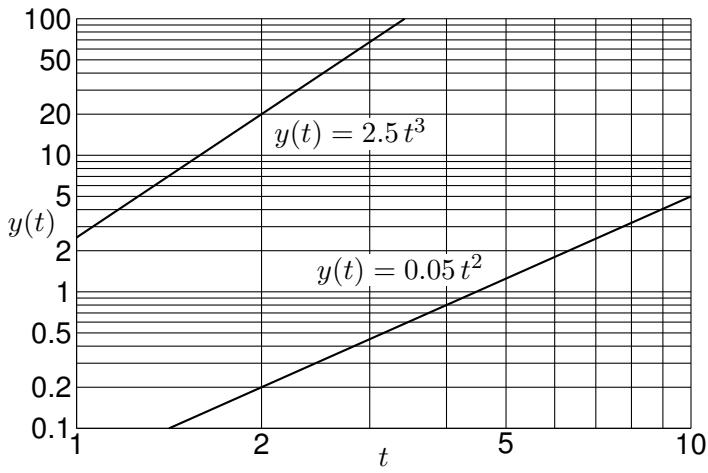

Figure 91: A graph and crosshatching example drawn using the pic language `[Loglog.m4]`.
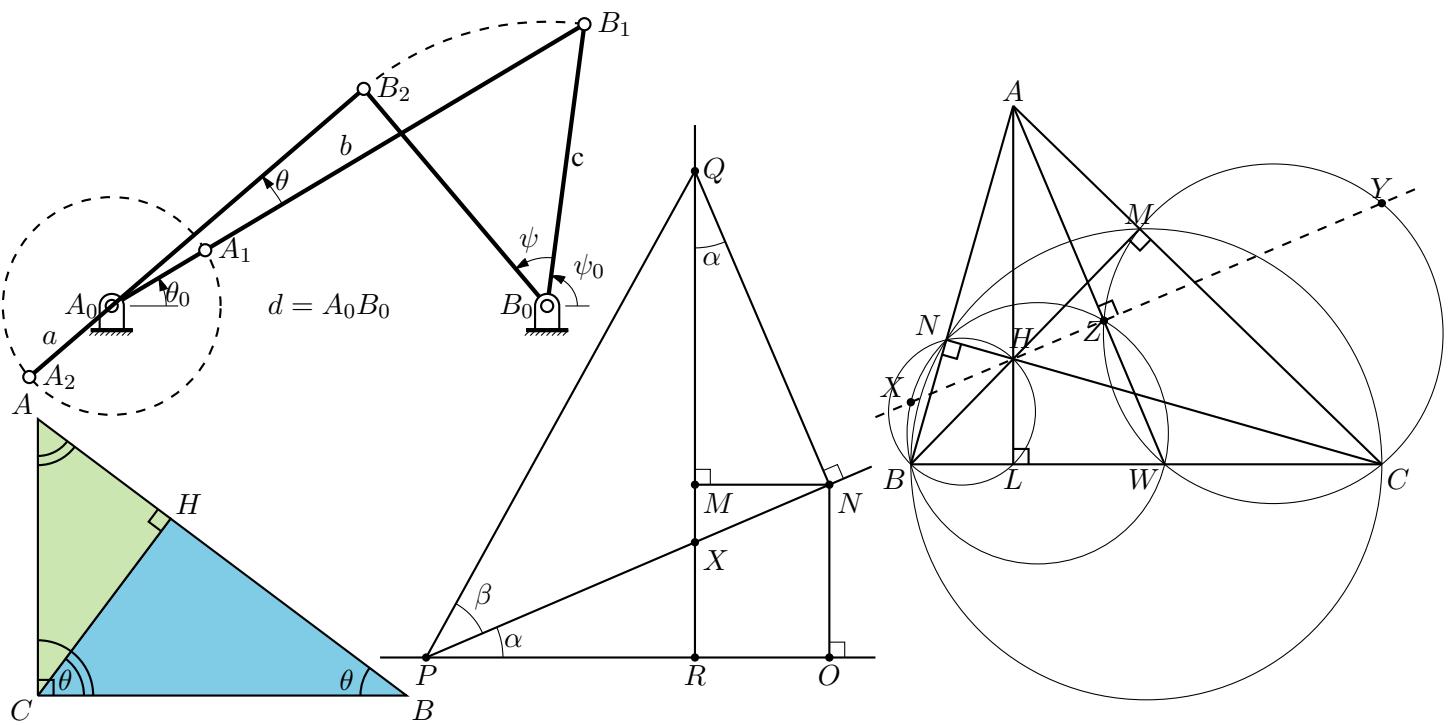
Figure 92: Some geometrical constructions `[Geometry.m4]`.
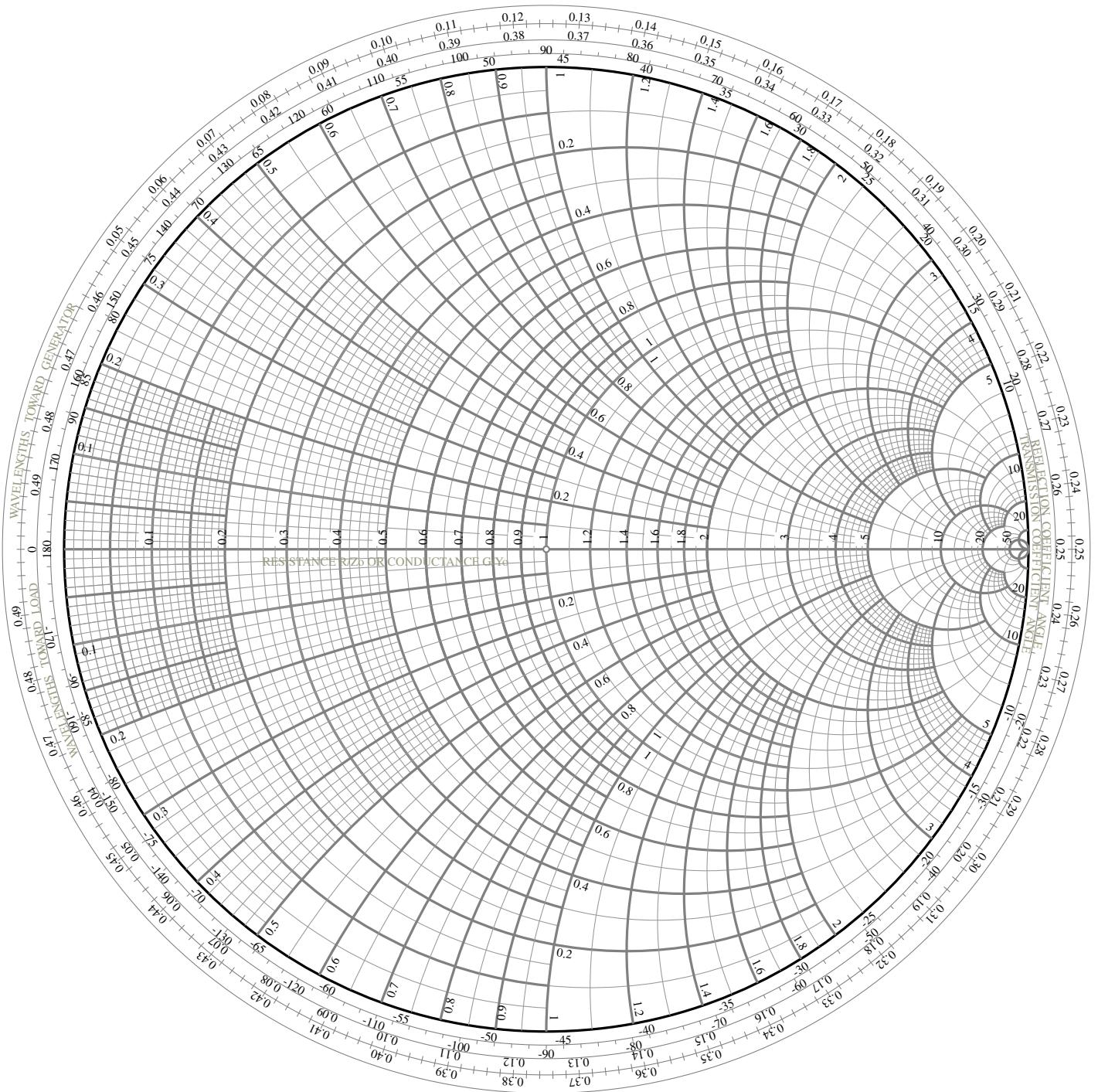
Figure 93: A Smith chart [Smithlchart.m4].

2.36 **4.22 (2) PL**

10.92 **27.76**

13.28
30.15
41.28

A AB B

CA X DB

C T^{15} D

C to D

blank width

label

offset *tic offset*

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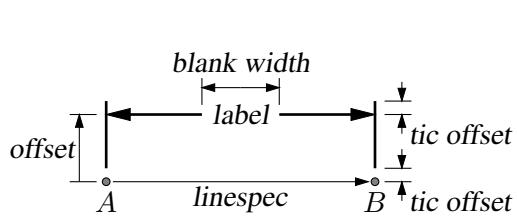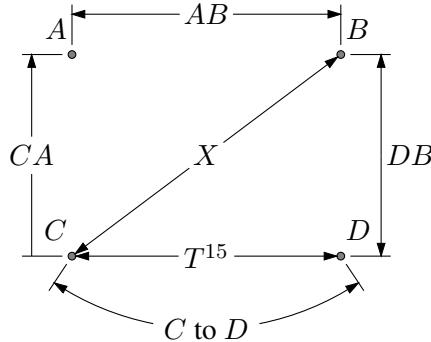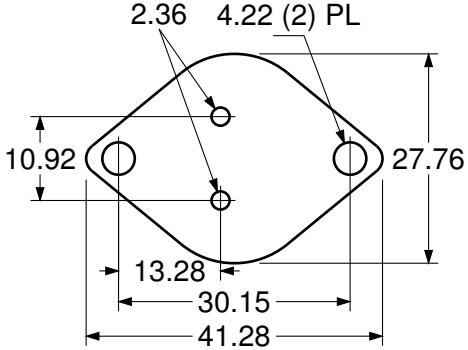
```
dimension_(from A to B,0.3,$AB$,20bp__)
dimension_(from C to B,,$X$,16bp__)
dimension_(from C to A,0.3,$CA$,14bp__,,->)
dimension_(from D to B,-0.3,"$DB$" ljust)
dimension_(from C to D,,s_box($T^{%g}$,15),W)
arcdimension_(from C to D with .c at \
   0.5 between A and B,12bp__,s_box($C$ to $D$),W)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

```
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]`.
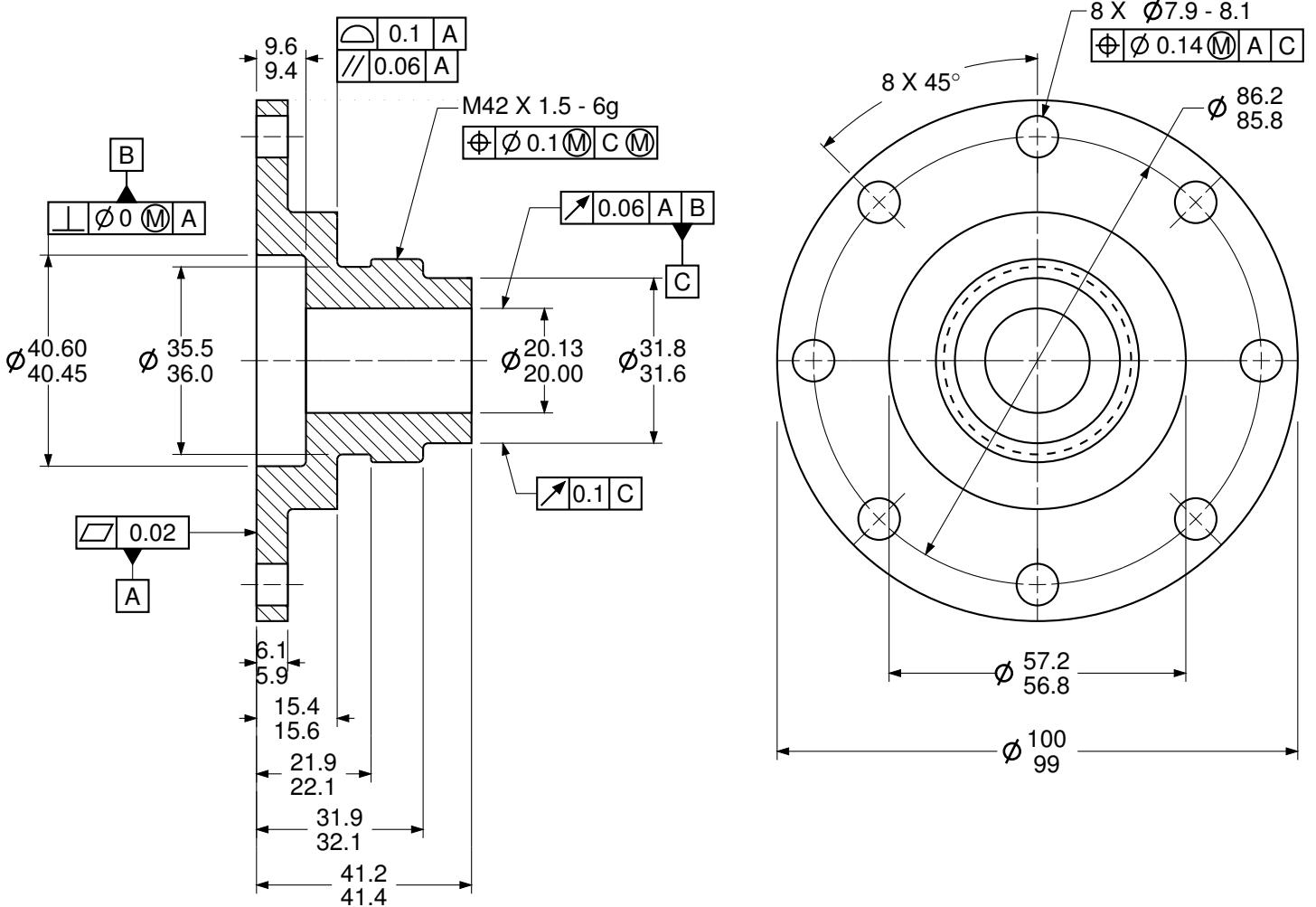
Figure 95: Dimensioning with tolerances [`Plate.m4`].
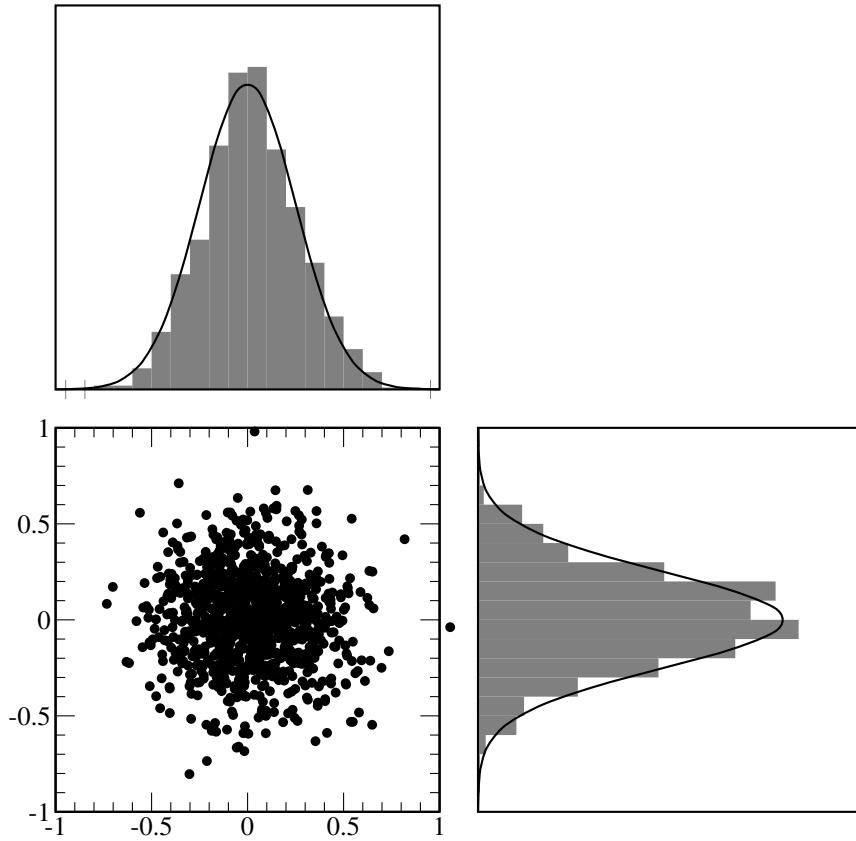
Figure 96: Testing random number generation using dpic macro randn (*array name, n, mean, std dev*) which calls pic built-in rand() [random.m4].
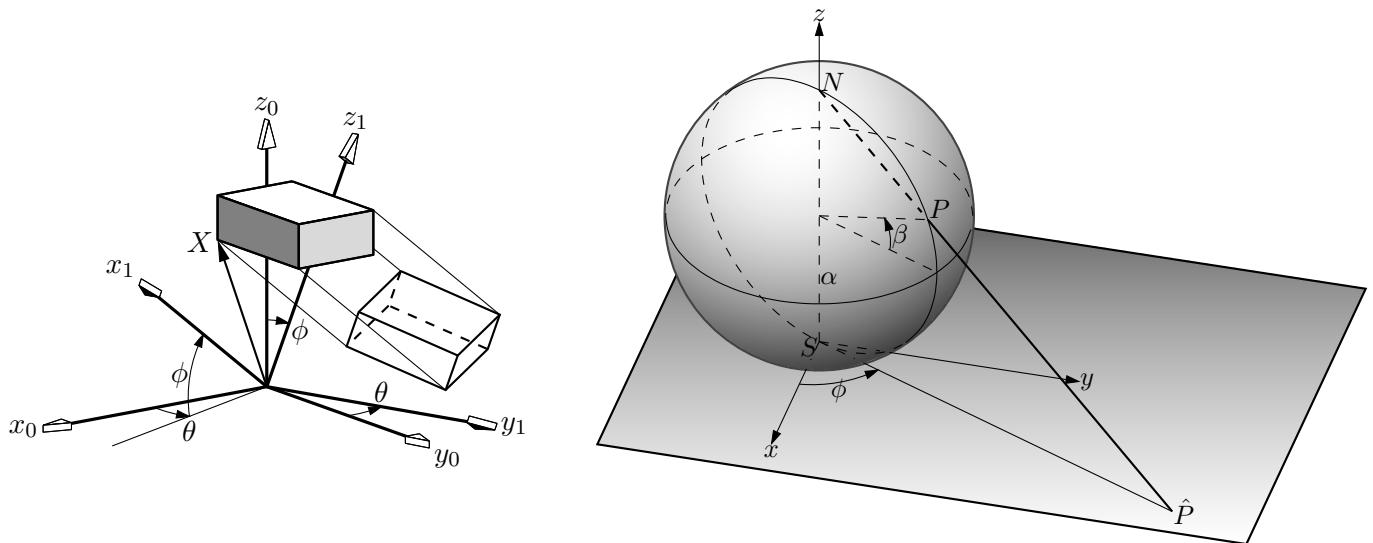

Figure 97: Test of project and other lib3D macros, showing the projection of a solid onto the y_1, z_1 plane by sighting along the x_1 axis. [exp.m4].
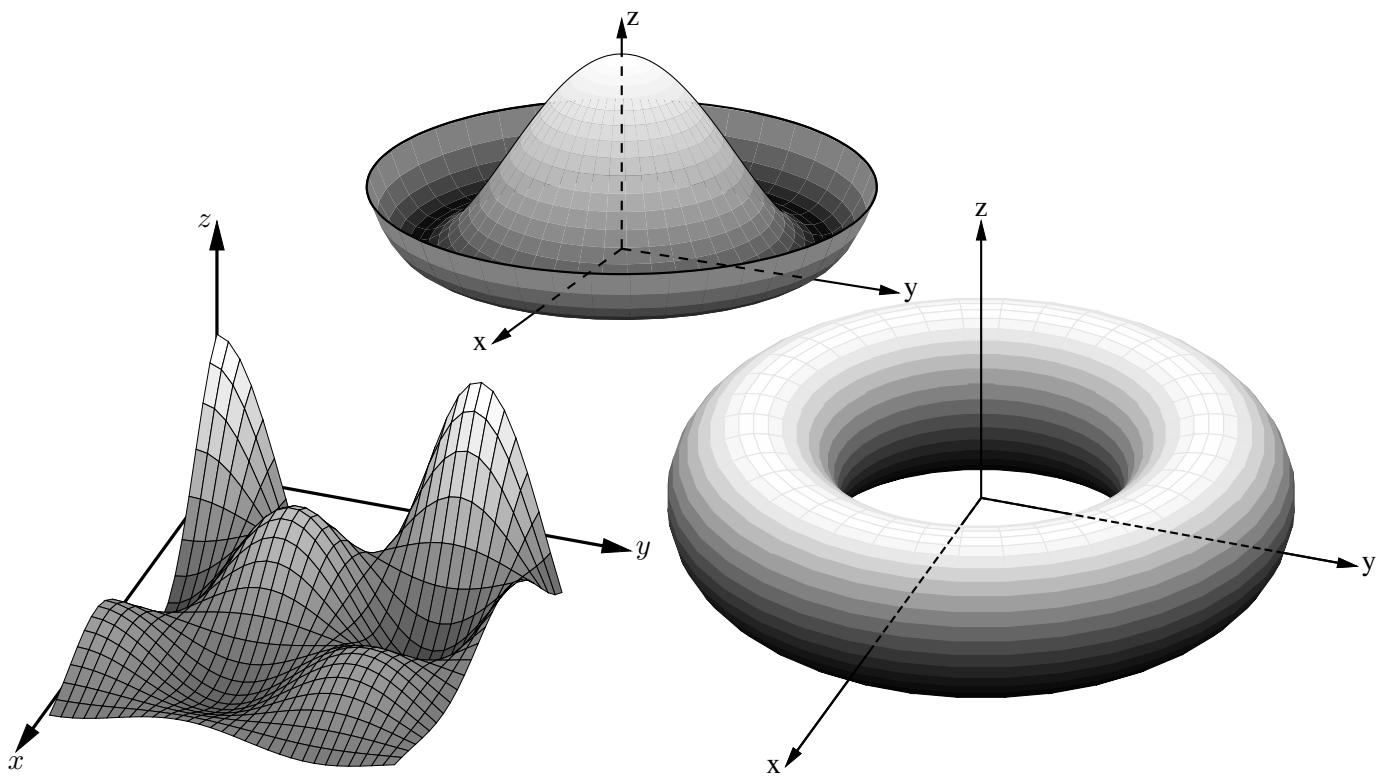
Figure 98: Plotting surfaces using gray scales [graysurf.m4].
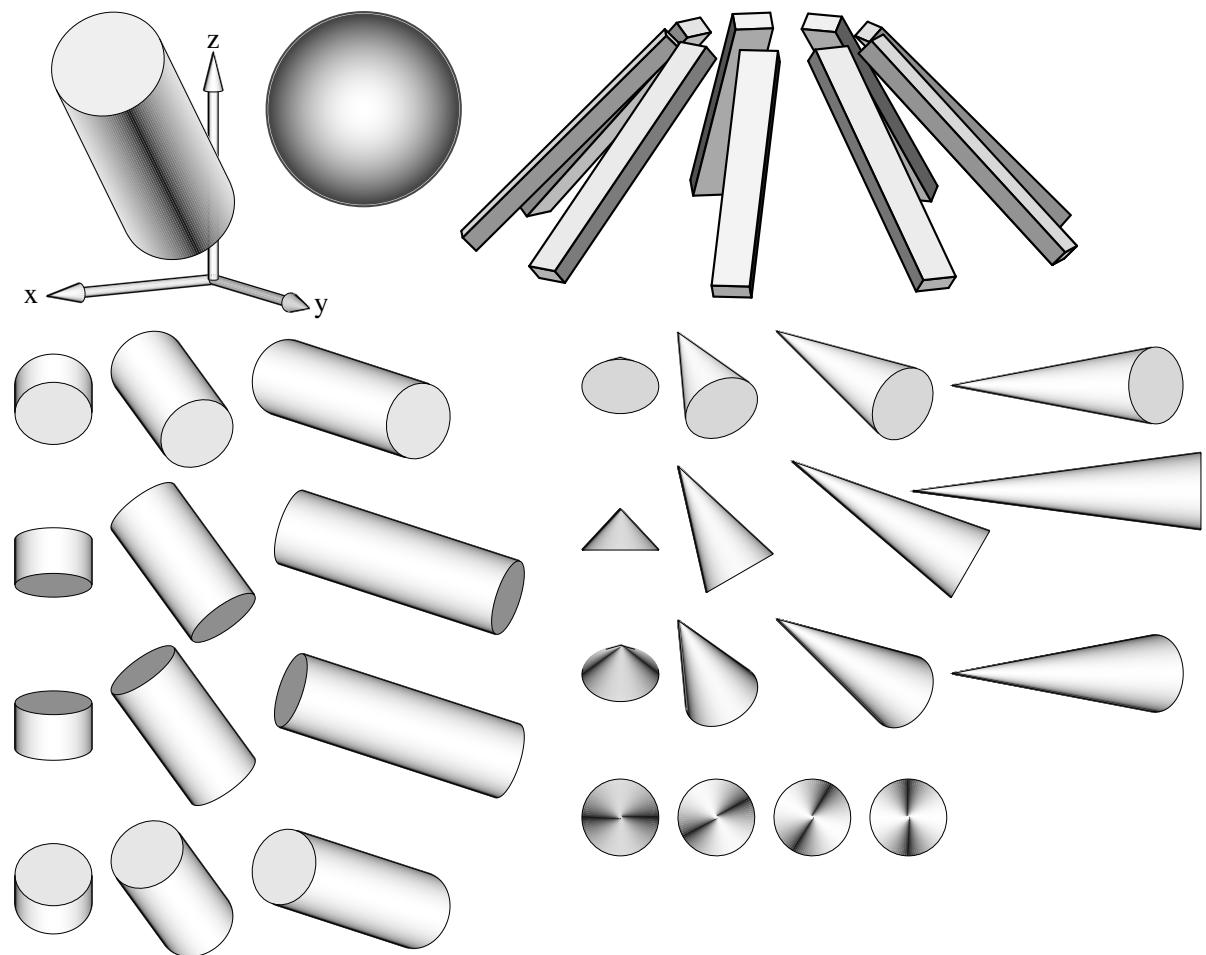
Figure 99: Basic shapes [shapes.m4].
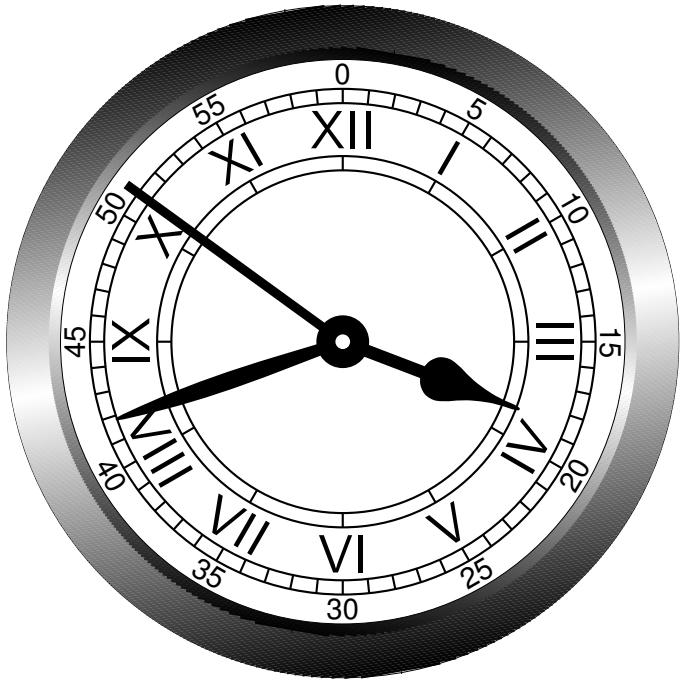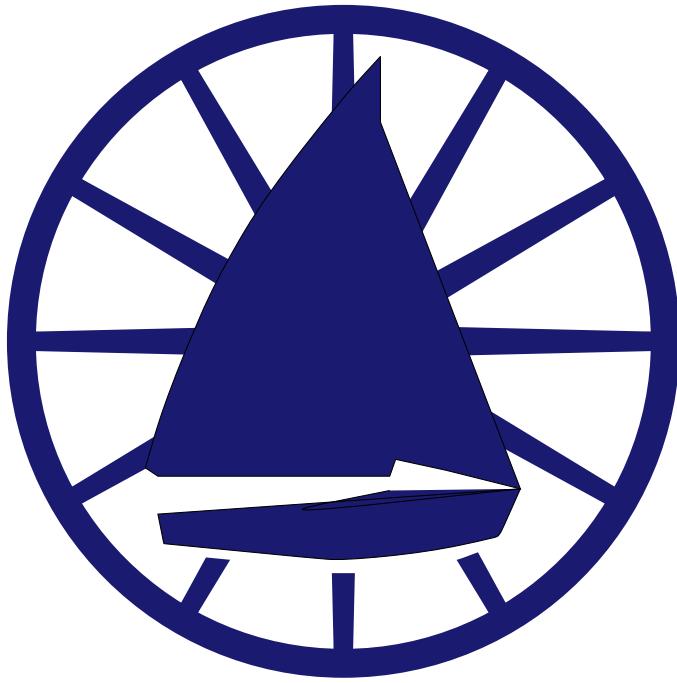
Figure 100: Conestoga Sailing Club (illustrating the filling of arbitrary shapes) and an antique clock face with shading and rotated text [csc.m4].
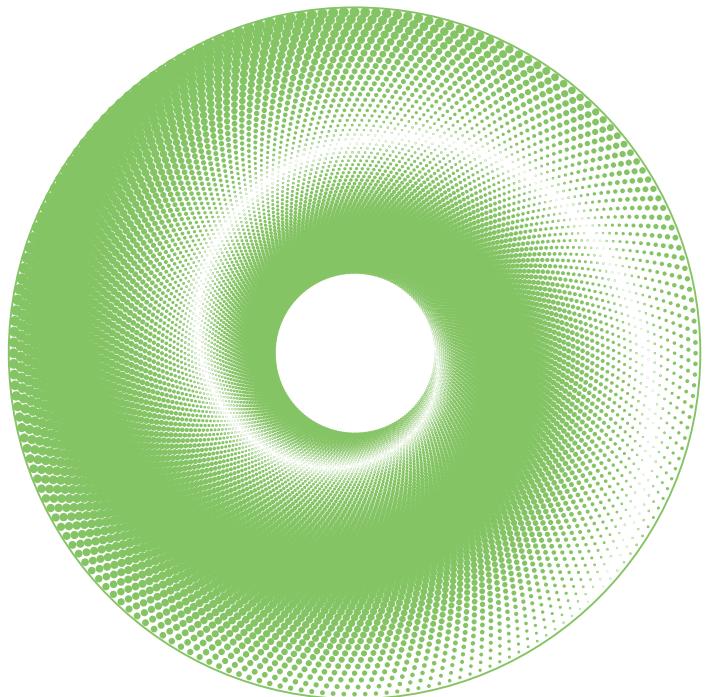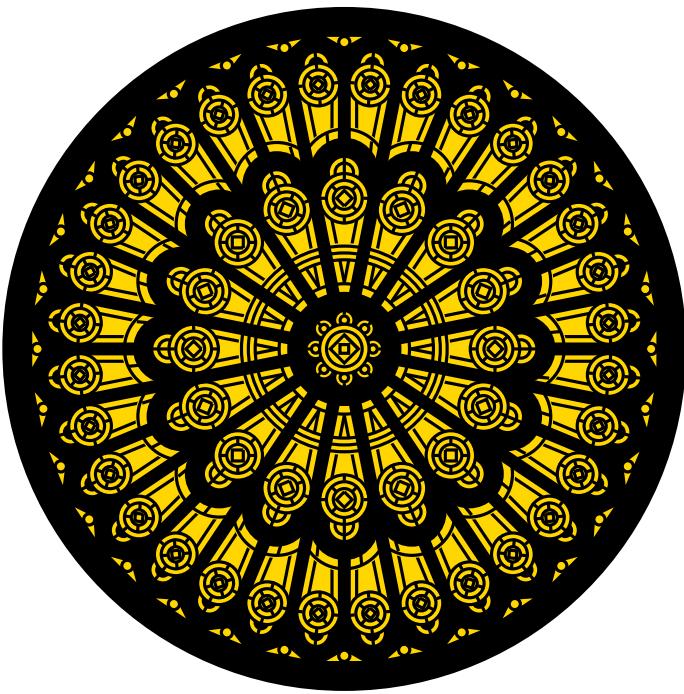

Figure 101: The left object, used for testing dipic, is redrawn from a detail of the set design for the musical *Dracula*. This consumes much LATEX main memory but can be produced directly as pdf using dpic -d, as svg using dpic -v, or as postscript using dpic -r since no text formatting is required. The right object adjusts the size of dots to produce a halftone effect [rose.m4].
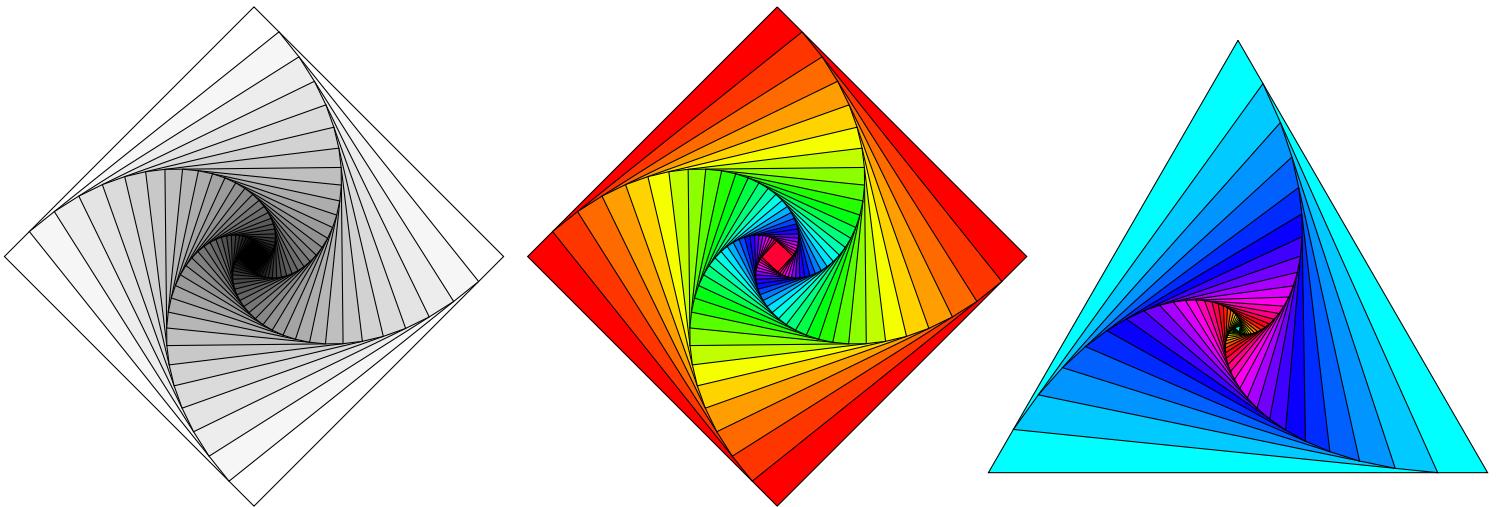
Figure 102: Variations on M. Goossens, S. Rahtz, and F. Mittelbach, *The LaTeX Graphics Companion,* Addison-Wesley 1997, pp. 57-58 [diamond.m4].
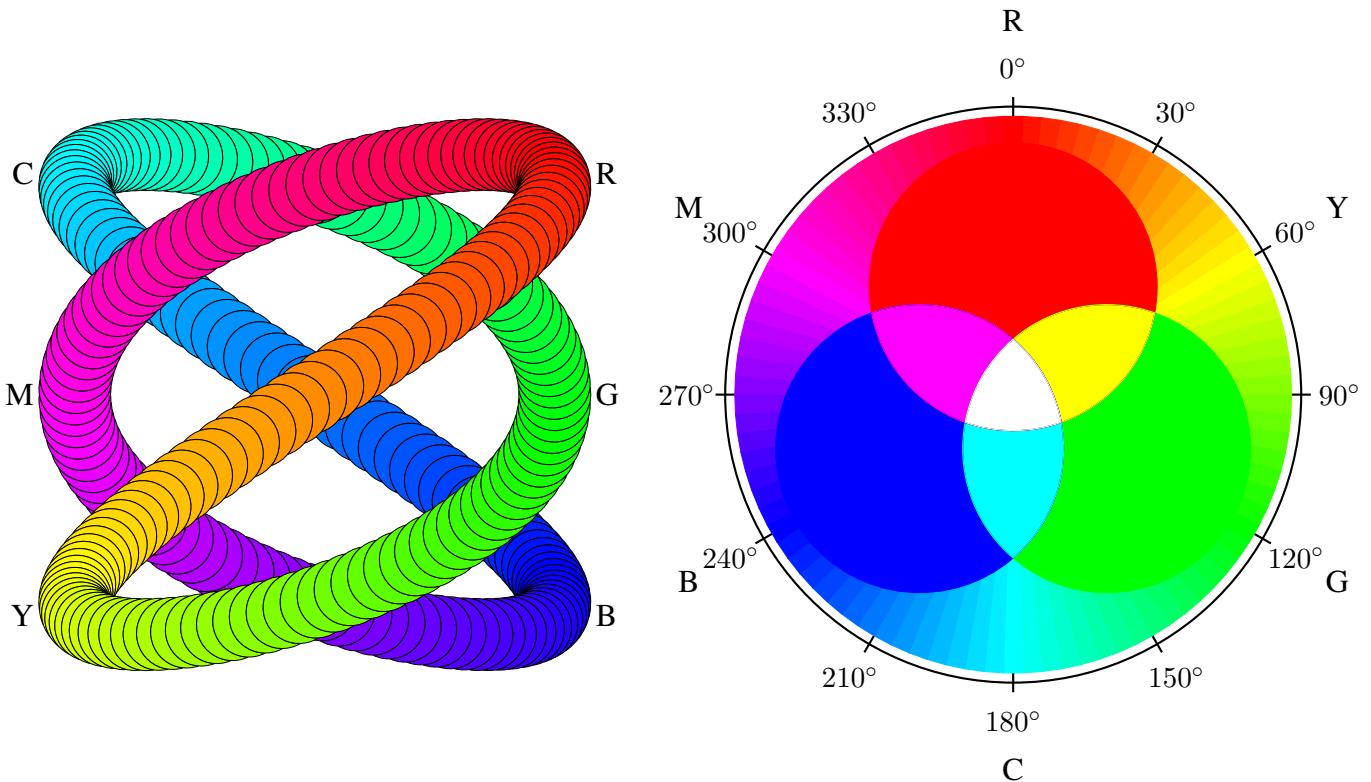

Figure 103: An exercise in calculating RGB colours [worm.m4].
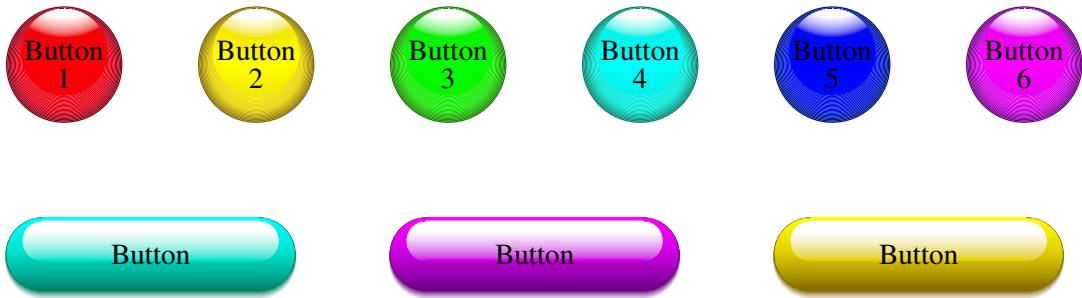
Figure 104: Shading in color [Buttons.m4].
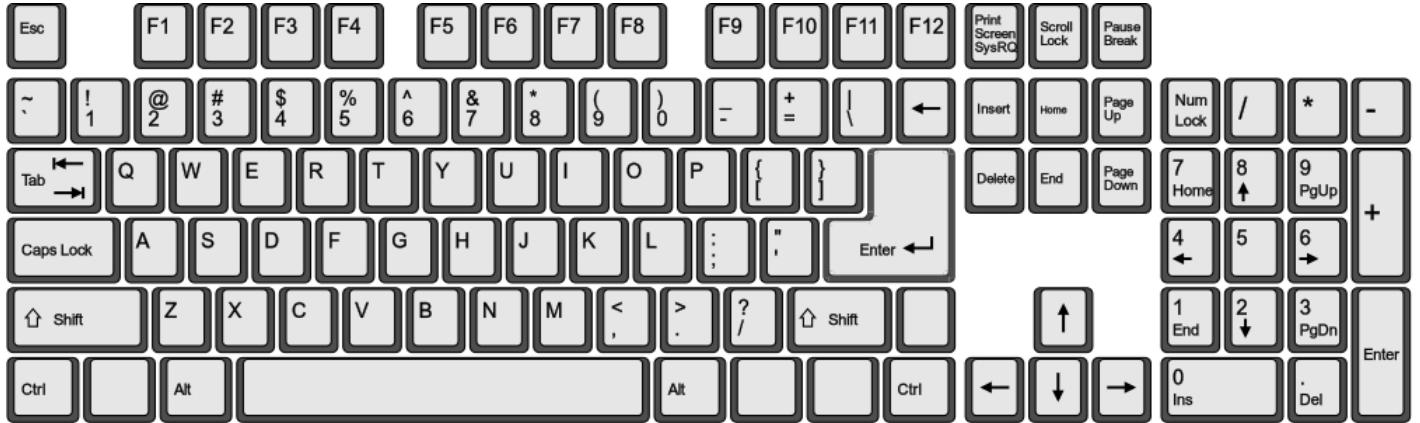
Figure 105: This diagram has been produced as svg with dpic -v (then converted to pdf for inclusion in examples.pdf) [keyboard.m4].
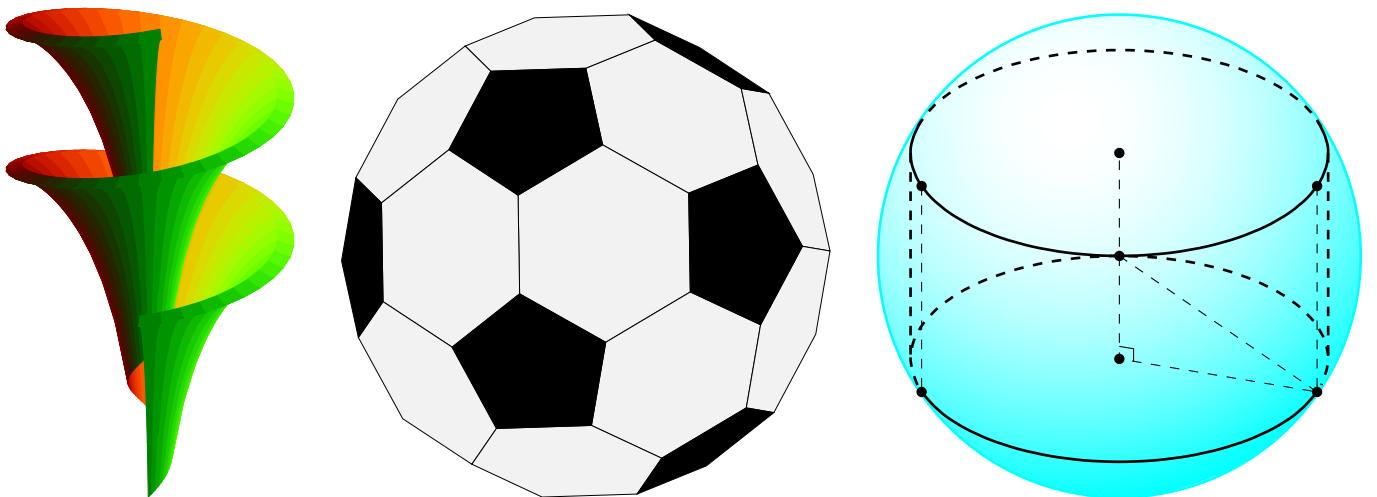

Figure 106: Dini surface, an icosahedron, and a sphere with inscribed cylinder [Dini.m4].

48
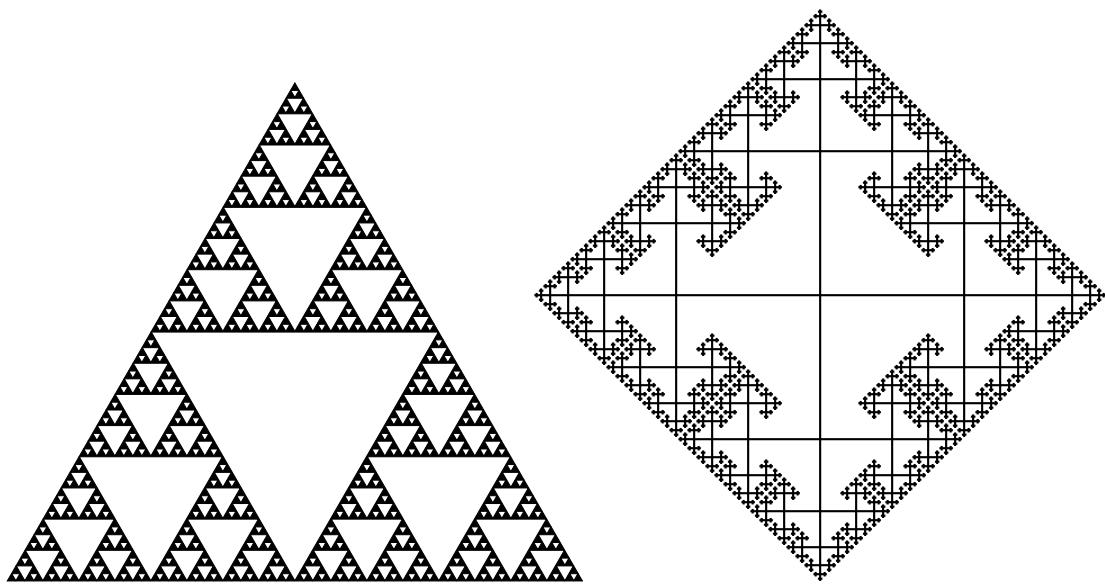
Figure 107: The Sierpinski triangle and a Cayley graph: tests of pic macro recursion [Sierpinski.m4].
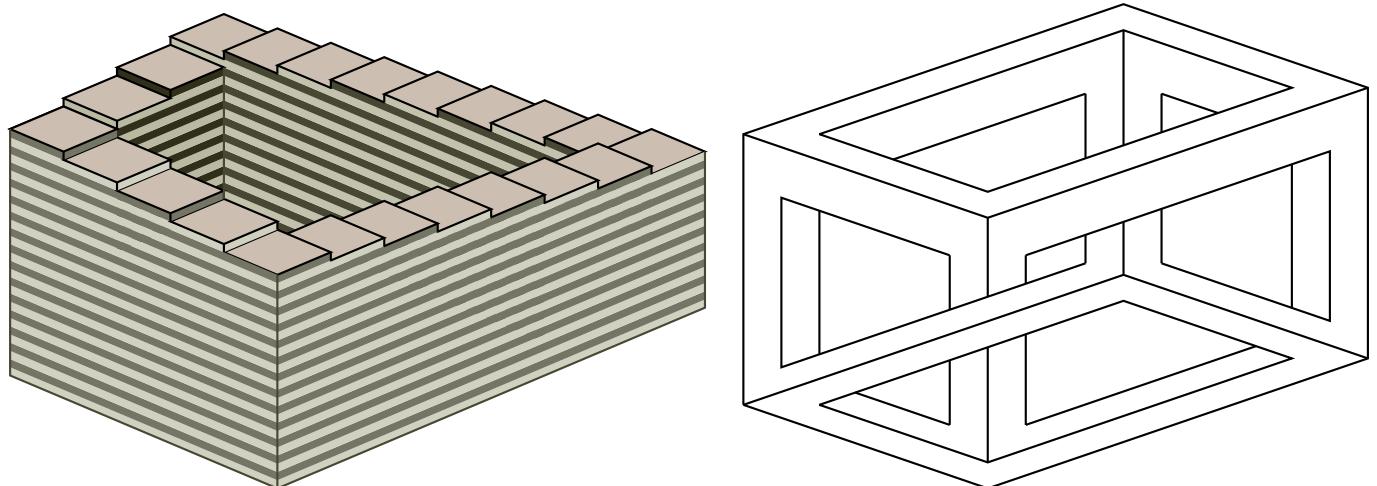

Figure 108: Penrose stairs and an Escher-like object [Escher.m4].
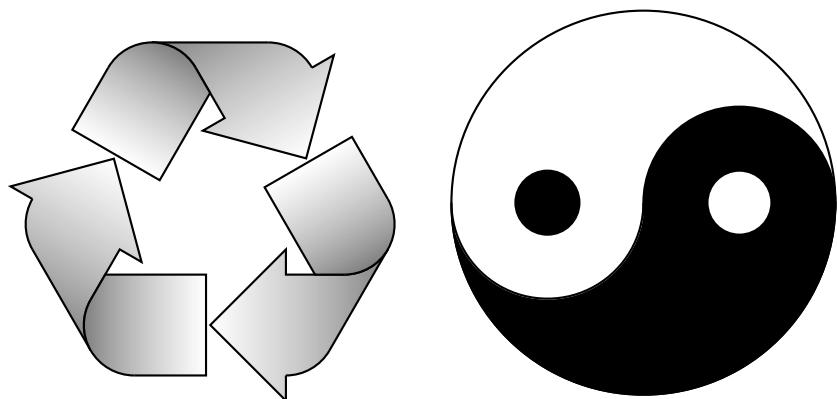
Figure 109: Modest repetition and partial fill [recycle.m4].
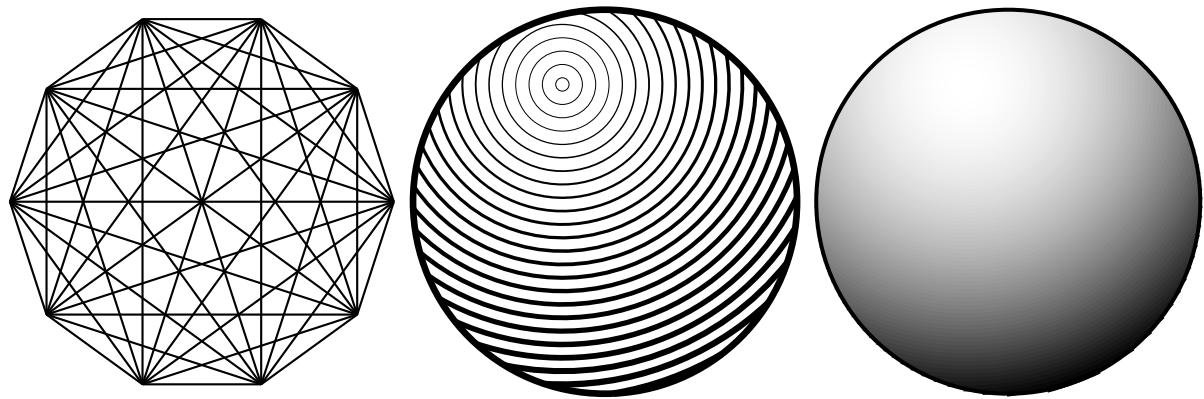
Figure 110: Simple diagrams that are easily drawn by looping `[ex15.m4]`.
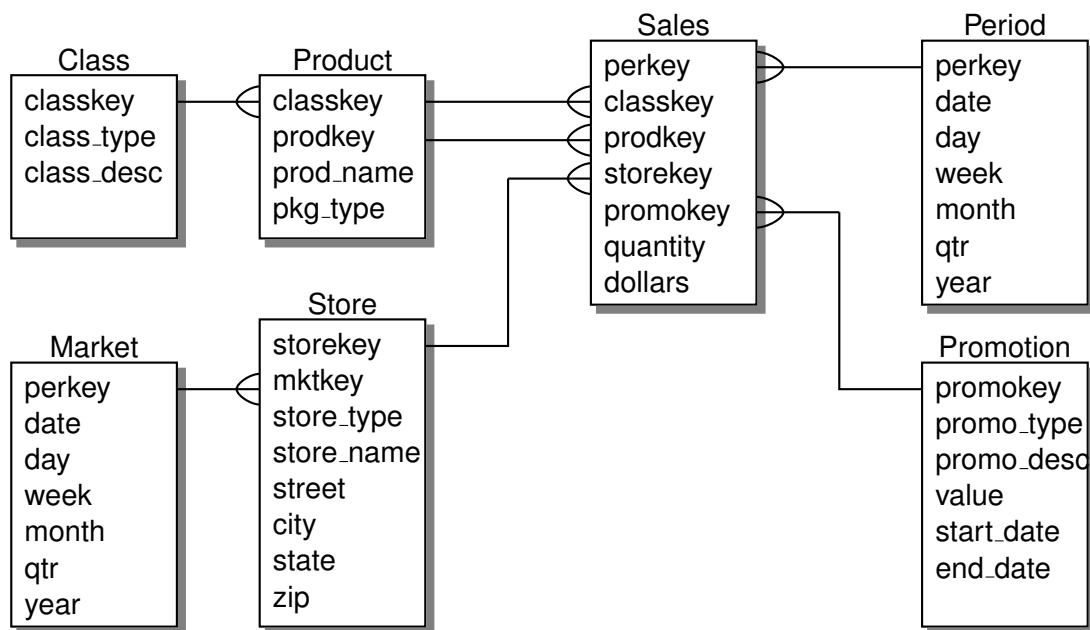
Figure 111: Illustrating `shadebox` and a custom crowfoot line termination `[Crow.m4]`.
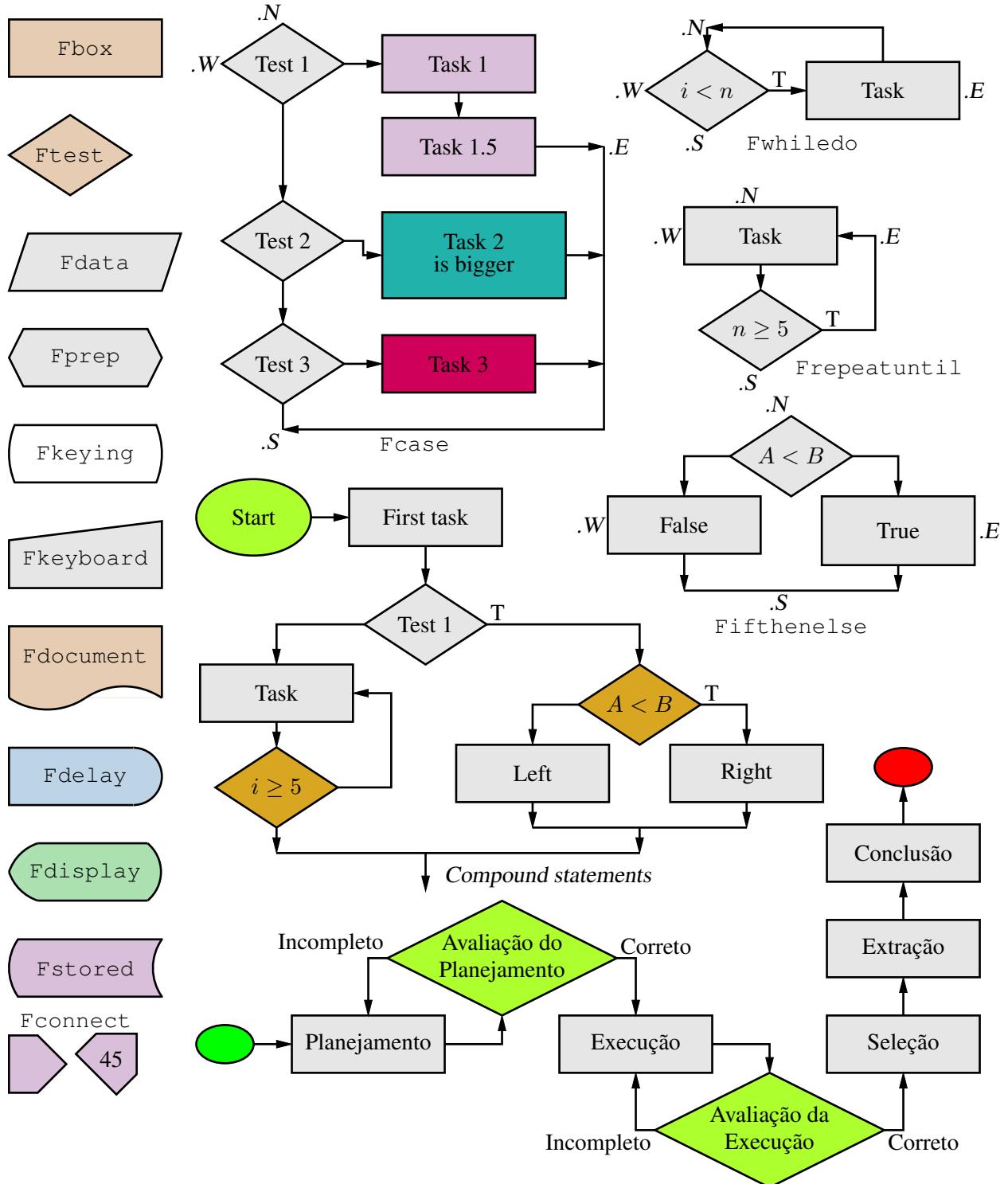
Figure 112: A flowchart sampler [Flow.m4].

The tree diagram contains the following nodes and labels:

Sirius — Canopus — Vega
Canopus — Alpha Centauri, Capella
Alpha Centauri — Achernar, Arcturus
Capella — Rigel
Arcturus — Betelgeux
Rigel — Procyon
Betelgeux — Beta Centauri

$h(h(h(x_1, x_2), h(x_3, x_4)), h(h(x_5, x_6), h(x_7, x_8)))$

$h(h(x_1, x_2), h(x_3, x_4))$

$h(h(x_5, x_6), h(x_7, x_8))$

$h(x_1, x_2)$

$h(x_3, x_4)$

$h(x_5, x_6)$

$h(x_9)$

$h(x_7, x_8)$

$x_1 := ((A, pk_A), h_1)$

$x_3 := ((A, pk'_A), h_3)$

$x_5 := ((E, pk_E), h_5)$

$x_7 := ((D, pk'_D), h_7)$

$x_2 := ((B, pk_B), h_2)$

$x_4 := ((D, pk_D), h_4)$

$x_6 := ((H, pk_H), h_6)$

$x_8 := ((K, pk_K), h_8)$

Figure 113: Trees [Btree.m4].

52

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