Examples: Version 9.6

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."
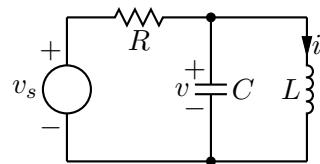

Figure 1: The quick-start example from the manual [`quick.m4`].
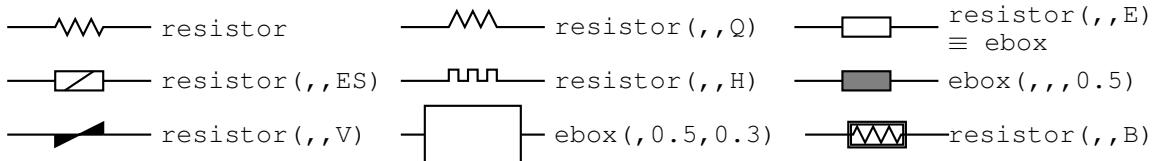

Figure 2: Resistors, showing some variations [`Resistors.m4`].
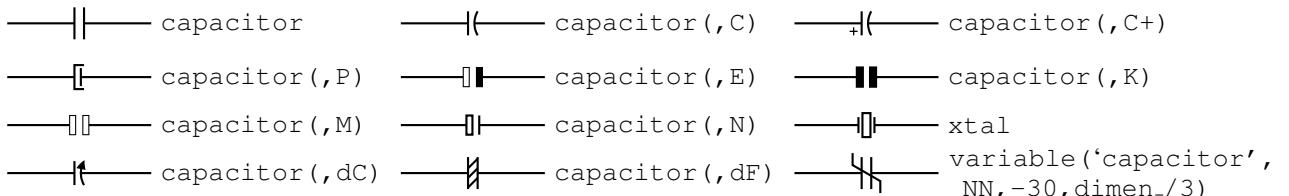
Figure 3: Capacitors, showing some variations [`Capacitors.m4`].
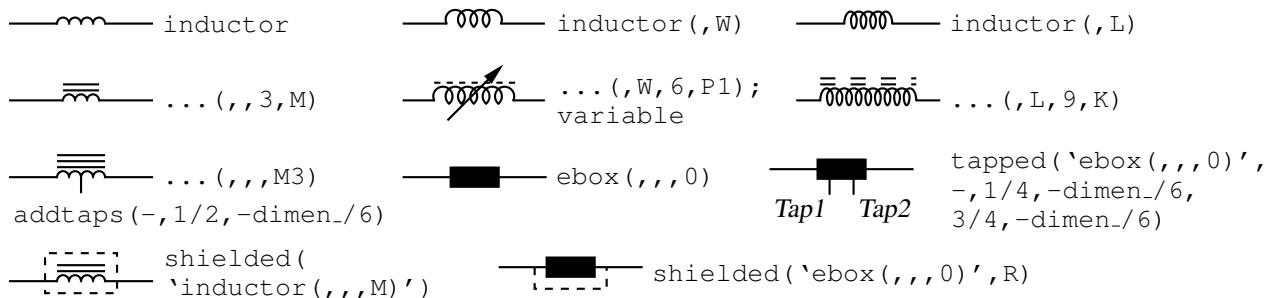
Figure 4: Inductors, showing some variations [`Inductors.m4`].

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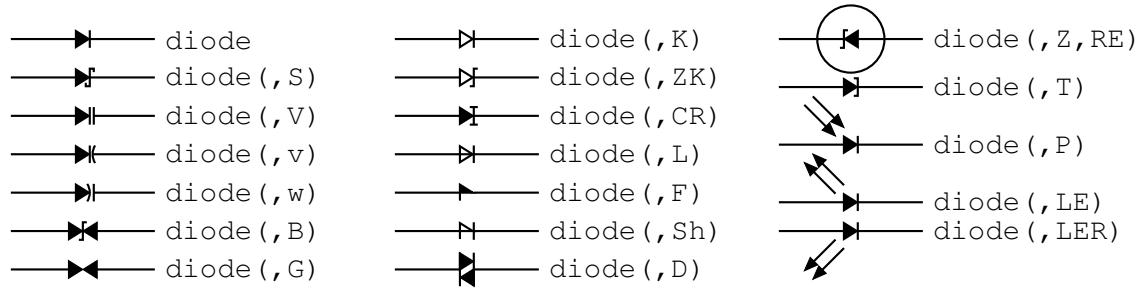
Figure 5: Diodes: a `K` in the second argument draws an open arrowhead [`Diodes.m4`].

Head A2
A1
Tail
em_arrows(N)

em_arrows(ND,45)   ...(I)   ...(ID)   ...(E)   ...(ED)
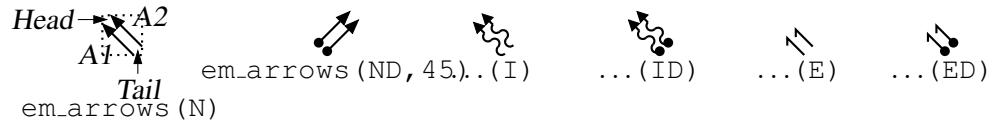
Figure 6: Radiation arrows [`Emarrows.m4`].

|		
source	source(,,0.4)	source(,N)
source(,I)	source(,P)	source(,"mA")
source(,i)	source(,U)	consource
source(,V)	source(,H)	consource(,I)
source(,v)	source(,R)	consource(,i)
source(,SC)	source(,S)	source(,SE)
source(,AC)	source(,SCr)	consource(,V)
source(,X)	source(,T)	consource(,v)
source(,F)	source(,L)	battery
source(,G)	source(,B)	battery(,3,R)
source(,Q)	nullator	
proximity = consource(,P)	norator	
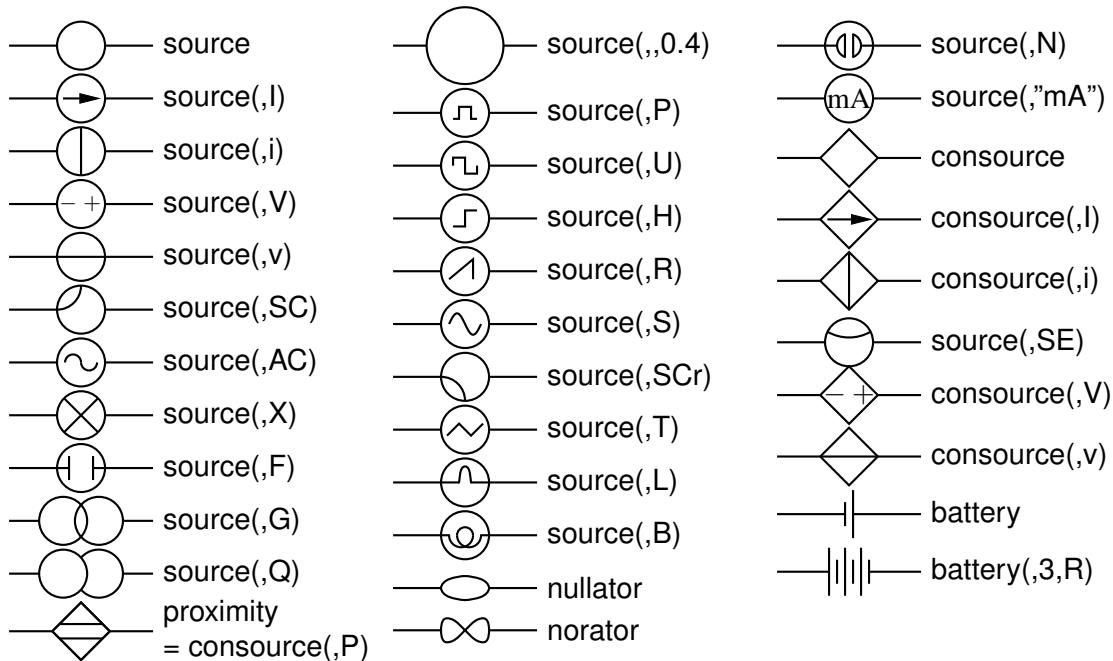
Figure 7: Sources and source-like elements [`Sources.m4`].
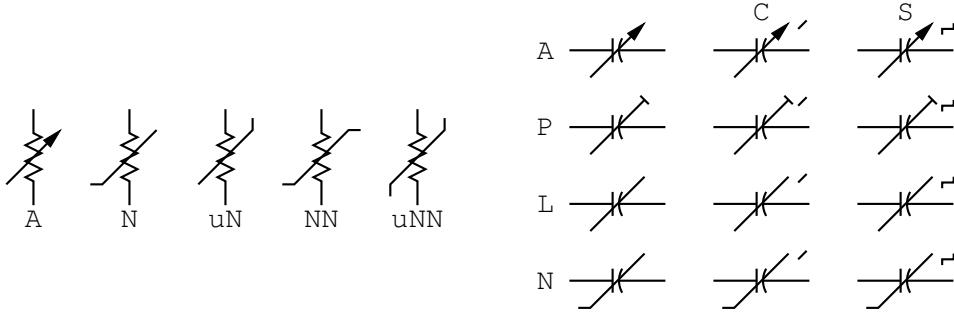
Figure 8: Arrows and marks indicating variability [Variable.m4].
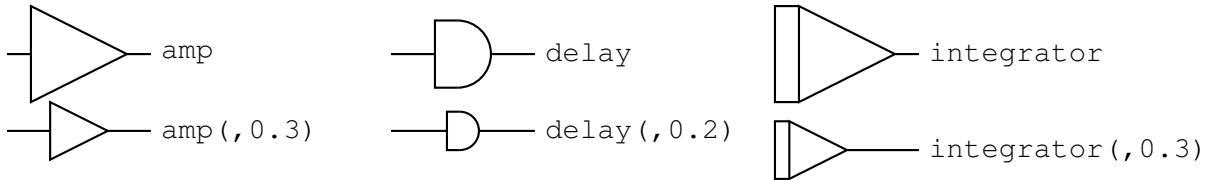
Figure 9: Macros amp, delay, and integrator [AmpTable.m4].
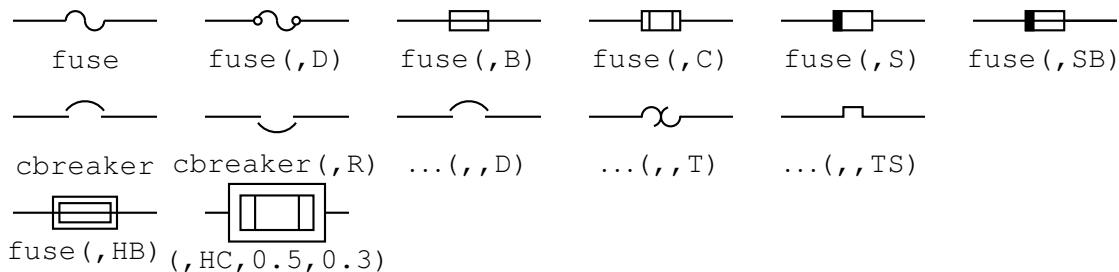
Figure 10: Macros fuse and cbreaker [Fuses.m4].
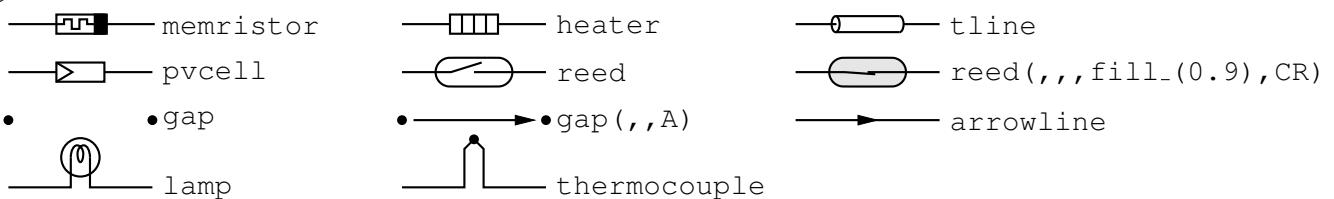
Figure 11: Additional two-terminal elements [MoreTable.m4].
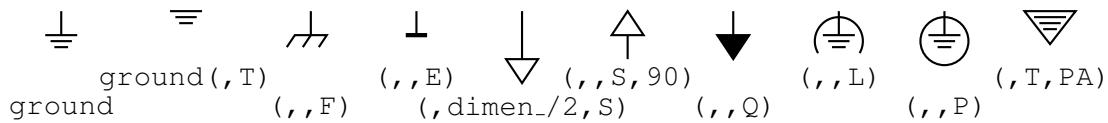

Figure 12: Ground symbols [Grounds.m4].

3
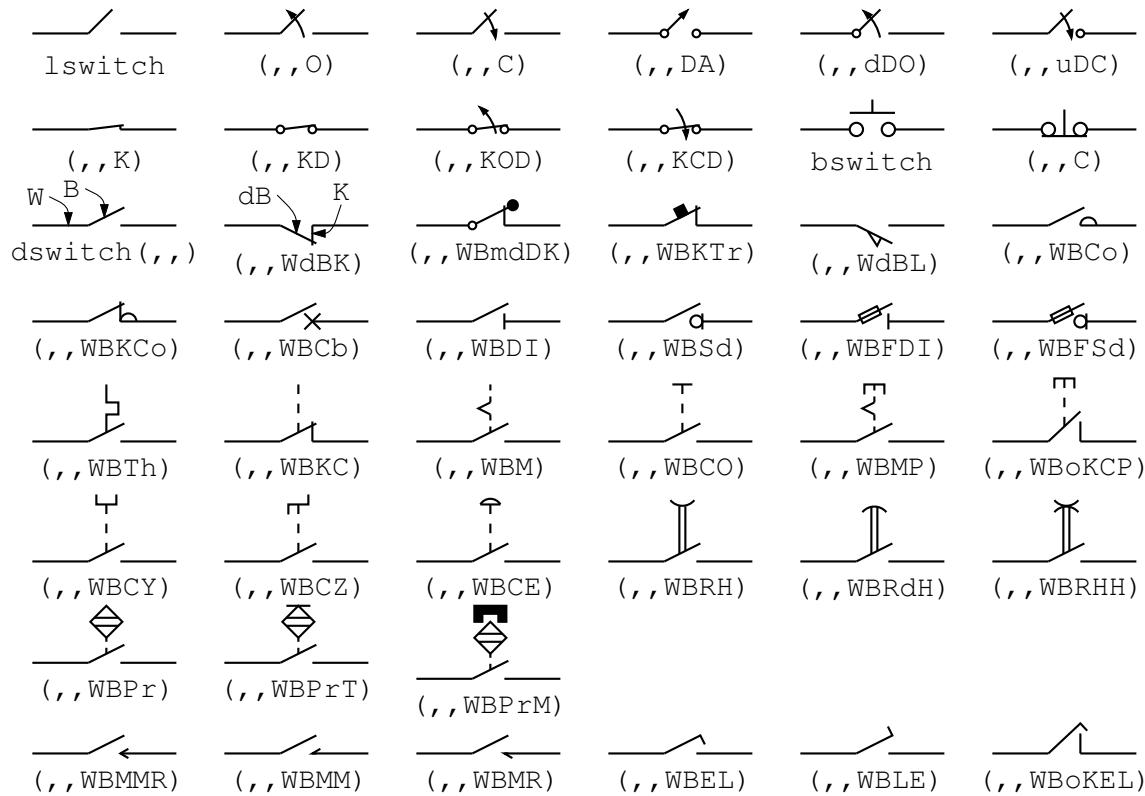
Figure 13: The switch macros; switch(,,,L|B|D) is a wrapper for lswitch, bswitch, and dswitch [Switches.m4].
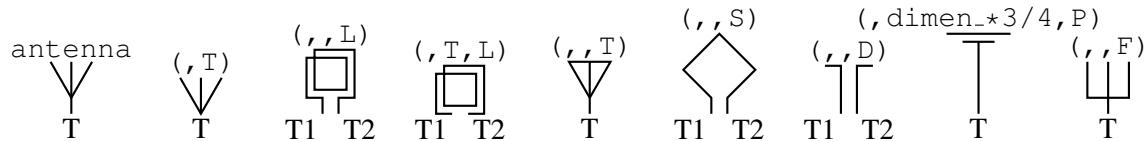
Figure 14: Antenna symbols [Antennas.m4].
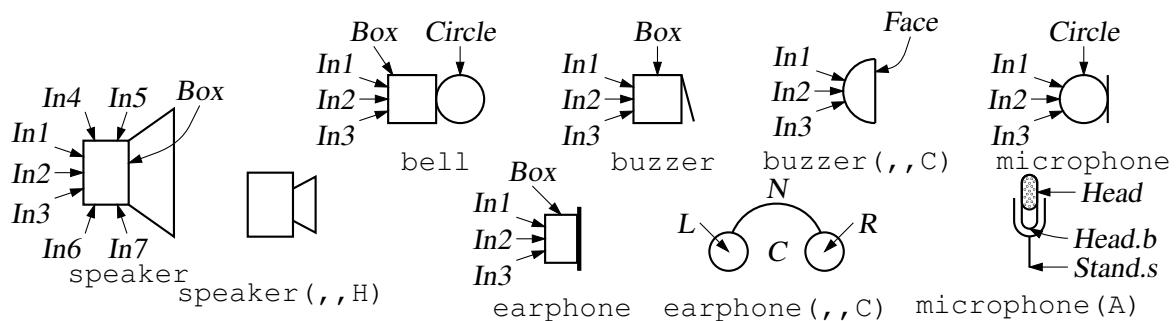

Figure 15: Audio elements [Audio.m4].
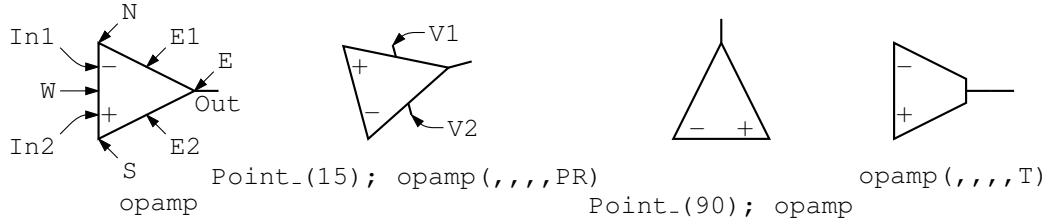
Figure 16: The opamp [`Opamp.m4`].
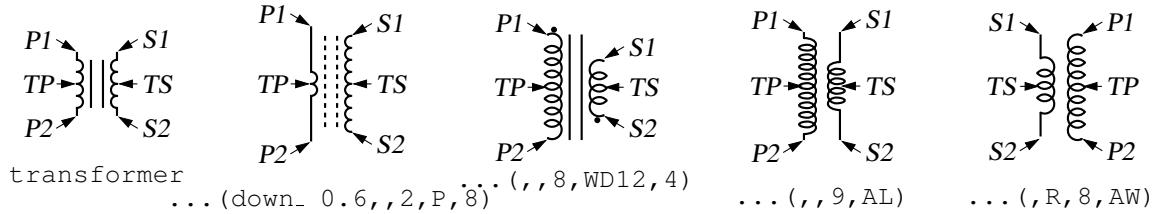
Figure 17: The transformer element, drawing direction down [`Xform.m4`].
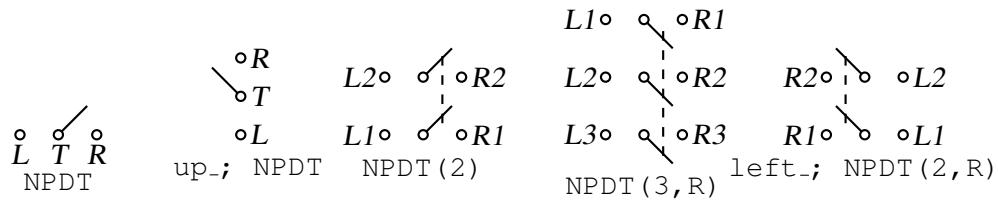

Figure 18: Double throw with the NPDT macro [`NPDT.m4`].
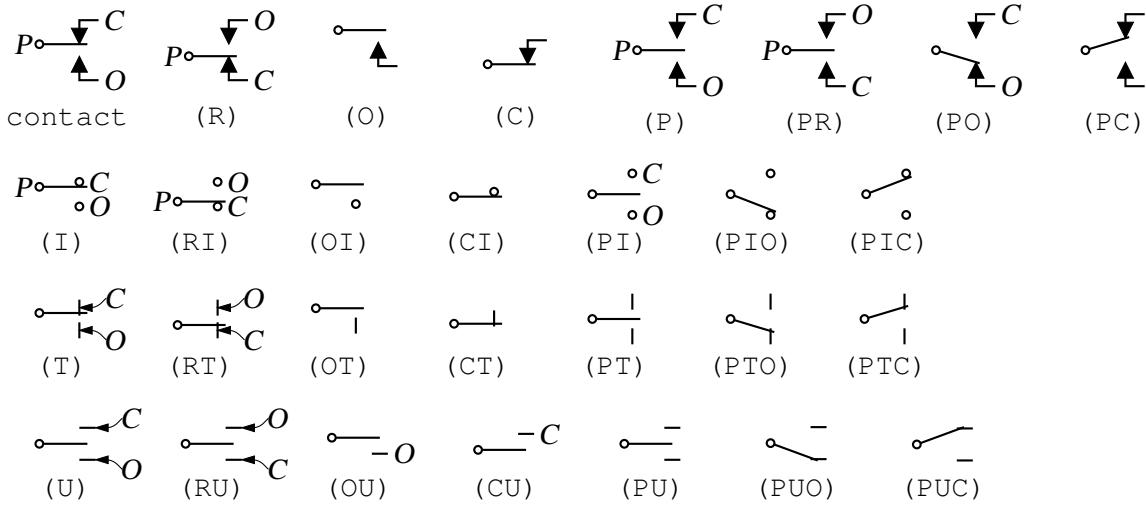
Figure 19: A non-exhaustive sampling of `contact` macro variations [`Contact.m4`].
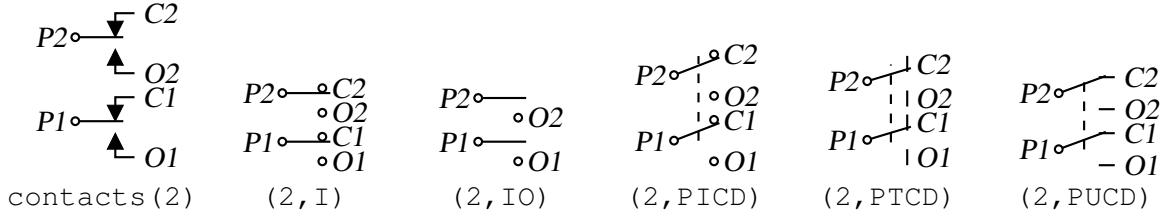
Figure 20: The `contacts` macro [`Contacts.m4`].
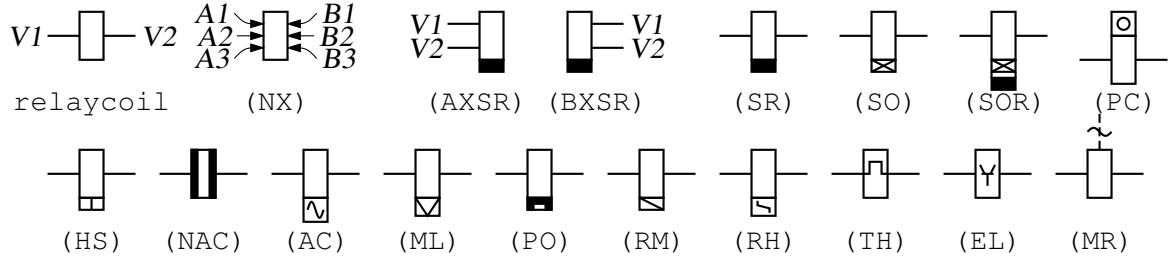

Figure 21: The `relaycoil` macro [`relaycoil.m4`].
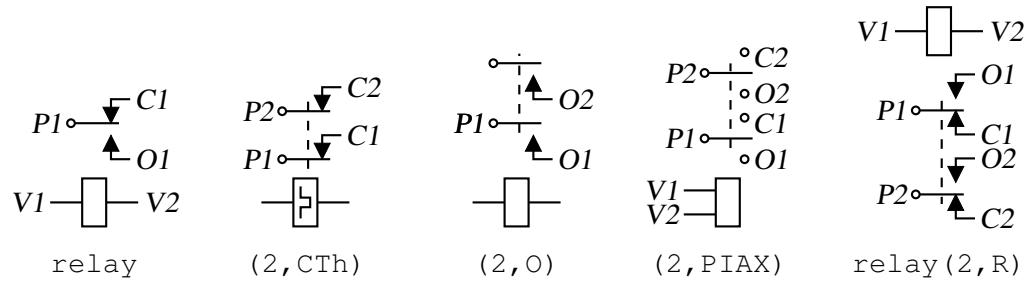

Figure 22: Some variants of the `relay` macro [`Relay.m4`].
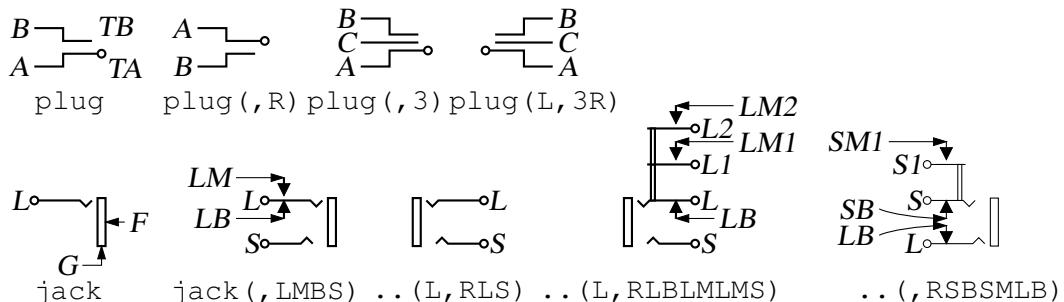

Figure 23: The `jack` and `plug` macros [`Jack.m4`].

6

L4 → T4 R4
⋮ ⋮
L1 → T1 R1

tstrip(U)

T1 ⋯ T5

tstrip(R,5,
DO;wid=1.0;ht=0.25)

ccoax(,F)

C
ccoax S

tconn(,<) (,<<)

tconn(,O) (,>) (,>>)

V_2
tbox(V_2)

V_1
tbox(V_1,,,<)

V_1
tbox(V_1,,,<>)
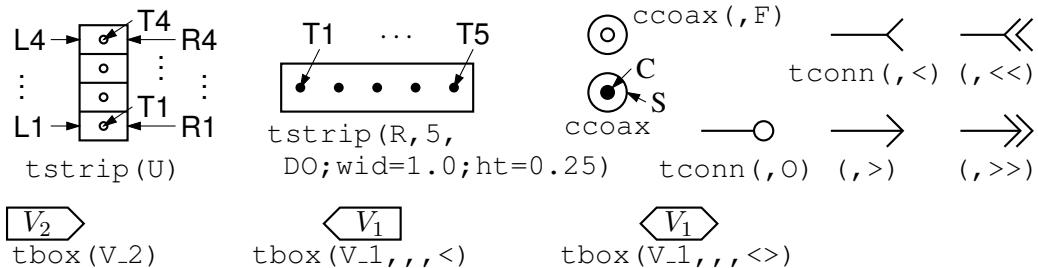
Figure 24: The tstrip, ccoax, tconn, and tbox macros [Conn.m4].

H   N   H

G

pconnex(,A) (,AF)   (,AC)   (,ACF)   (U,D) (U,DF) (U,J) (U,JF)

(,G)   (,GF)   (L,GF)   (,GC)   (,GCF)

(,P)   (,PF)
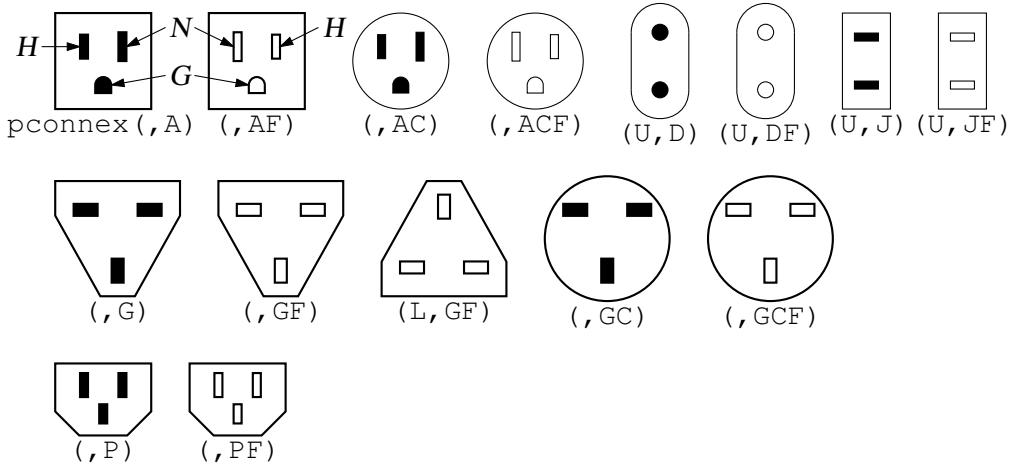
Figure 25: The pconnex macro [Pconn.m4].

P1   Block
P2

Header

P1   P2
P5   P6

Header(2,3,8mm__,10mm__)

P8   P1

left_; Header(2,4,,,fill_(0.9))

P15   P1
                     PinP1
                     PinP2
P16   P2

down_; Header(2,8)
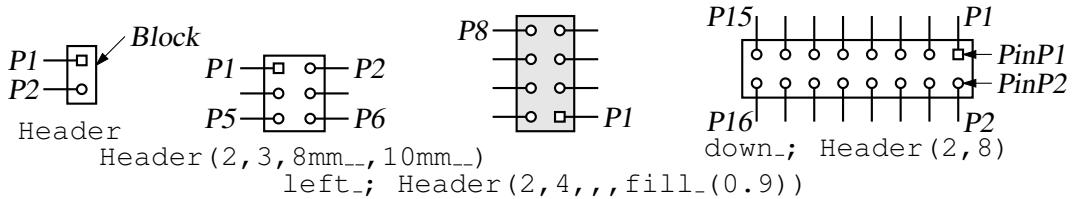
Figure 26: The Header macro [Headers.m4].

7

Arduino Leonardo

SCL, SDA, AREF, GND, ~D13, D12, ~D11, ~D10, ~D9, D8, D7, ~D6, ~D5, D4, ~D3, D2, TX D1, RX D0

IOREF, RESET, +3V3, +5V, GND, GND, VIN, A0, A1, A2, A3, A4, A5

Raspberry Pi 4 GPIO

3V3	5V
2 SDA1	5V
3 SCL1	GND
4 GPIO_GCLK	TXD0 14
GND	RXD0 15
17 GPIO_GEN0	GPIO_GEN1 18
27 GPIO_GEN2	GND
22 GPIO_GEN3	GPIO_GEN4 23
3V3	GPIO_GEN5 24
10 SPI_MOSI	GND
9 SPI_MISO	GPIO_GEN6 25
11 SPI_SCLK	SPI_CE0_N 8
GND	SPI_CE1_N 7
ID_SD	ID_SC
5	GND
6	12
13	
19	GND
26	16
GND	20
	21

Micro HDMI

20 SH1, 21 SH2
1 HPD, 2 UTILITY, 3 D2+, 4 D2_SH, 5 D2−, 6 D1+, 7 D1_SH, 8 D1−, 9 D0+, 10 D0_SH, 11 D0−, 12 CLK+, 13 CLK_SH, 14 CLK−, 15 CEC, 16 GND, 17 SCL, 18 SDA, 19 +5V
22 SH3, 23 SH4

USB-C Socket

B12 GND / GND A1
RX1+ / TX1+
RX1− / TX1−
VBUS / VBUS
SBU2 / CC1
D− / D+
D+ / D−
CC2 / SBU1
VBUS / VBUS
TX2− / RX2−
B1 TX2+ / RX2+ A12
GND / GND

Figure 27: Some connectors with simple geometry and lists of labels [`Connectors.m4`].

4017

1 PL	Vcc 16
2 Q3	CLK 15
3 I3	Q2 14
4 I0	I2 13
5 CLE	I1 12
6 Q0	Q1 11
7 TC	U/D 10
8 GND	MR 9

4510

1 PL	Vcc 16
2 Q3	CLK 15
3 I3	Q2 14
4 I0	I2 13
5 CLE	I1 12
6 Q0	Q1 11
7 TC	U/D 10
8 GND	MR 9

VS1053

1 LOUT	MIC+ 32
2 ROUT	MIC− 31
3 GBUF	AGND 30
4 AGND	3V3 29
5 AGND	GND 28
6 DREG	0 27
7 Vcc	1 26
8 3V3	2 25
9 GND	3 24
10 MISO	4 23
11 MOSI	5 22
12 sclk	6 21
13 RST	7 20
14 CS	TX 19
15 DCS	RX 18
16 DCS	SDCD 17

DIP_chip_outline

Body
Pin1 *Pin8*

DIP

Pin1

Pin32

DIP(32,direct=R;bodywid=0.5*scale)   DIP(16,type=Q;pinwid=1bp__)

Figure 28: IC package outlines [`Chips.m4`].

8
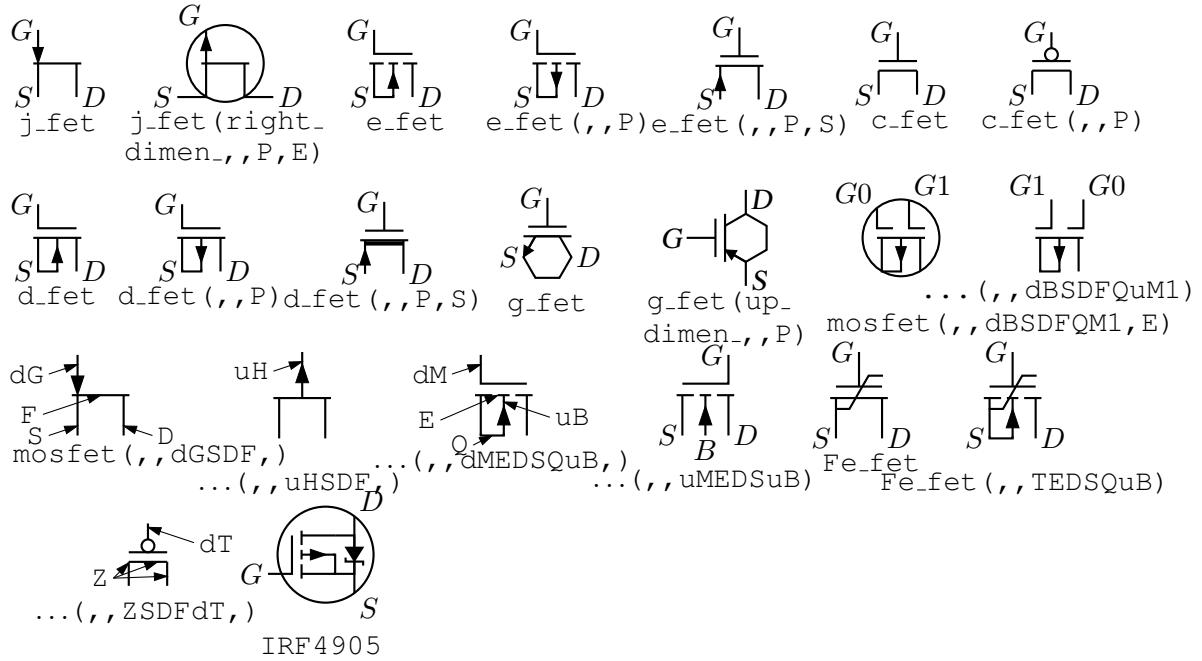
Figure 29: FETs, showing programmable components and example customizations [fet.m4].
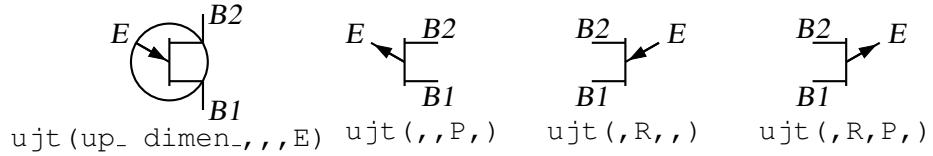

Figure 30: UJT examples [ujt.m4].
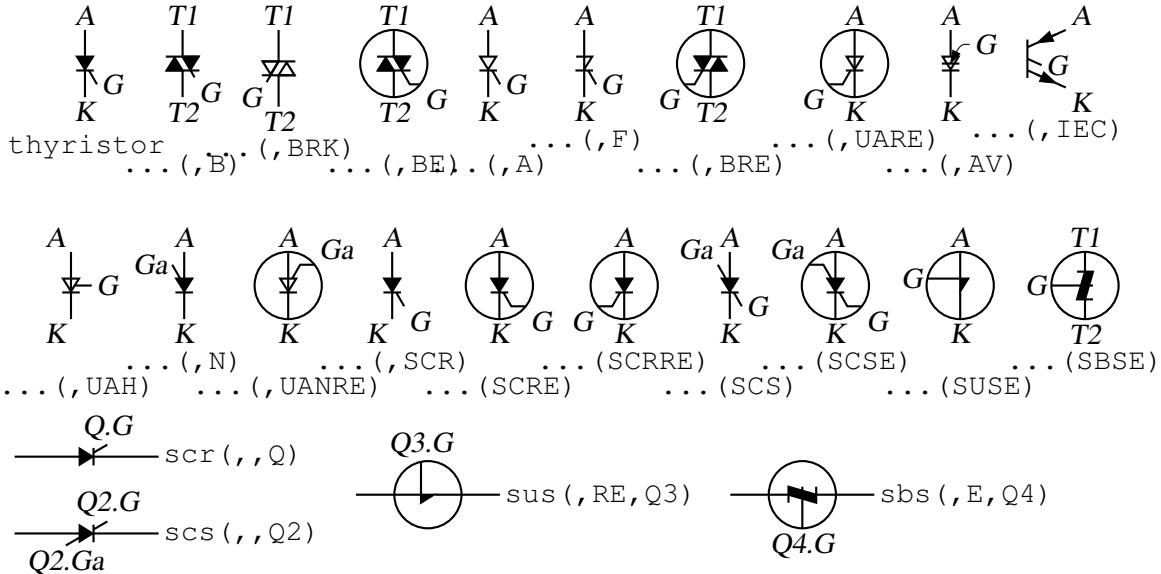

Figure 31: Thyristor examples [thyristor.m4].
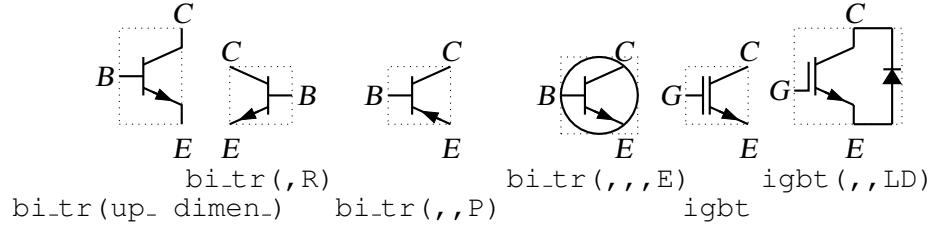
Figure 32: Bipolar transistors (drawing direction: up) [Bip.m4].
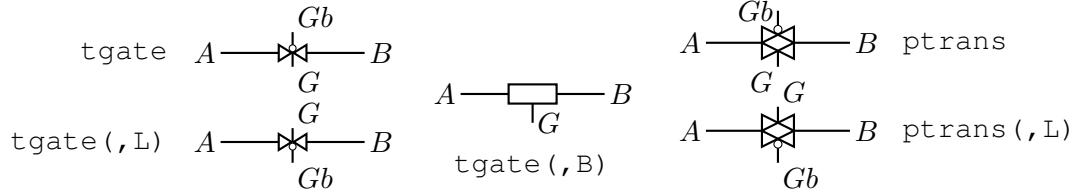
Figure 33: The tgate and ptrans elements [Tgate.m4].

nport(wid 2.0 ht 1 fill_(0.9) "n-port",1,2,3,4)
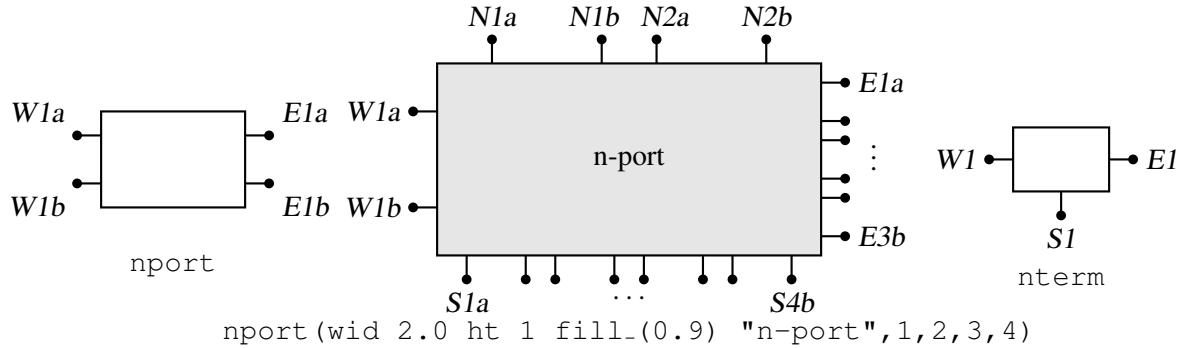
Figure 34: The nport and nterm macros [Nport.m4].
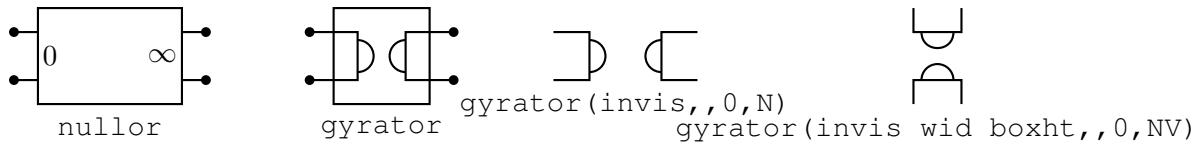
Figure 35: Some customizations of nport [NLG.m4].
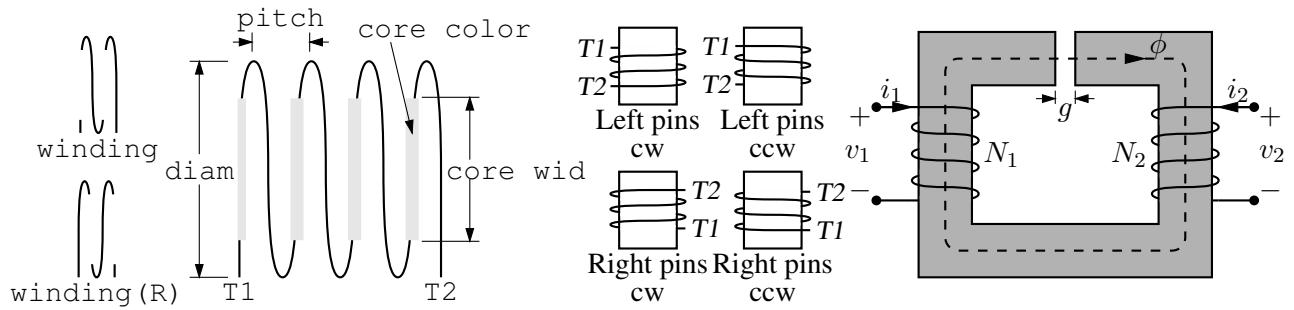
Figure 36: The macro `winding(L|R,diam,pitch,turns,core wid,core color)` [Windings.m4].
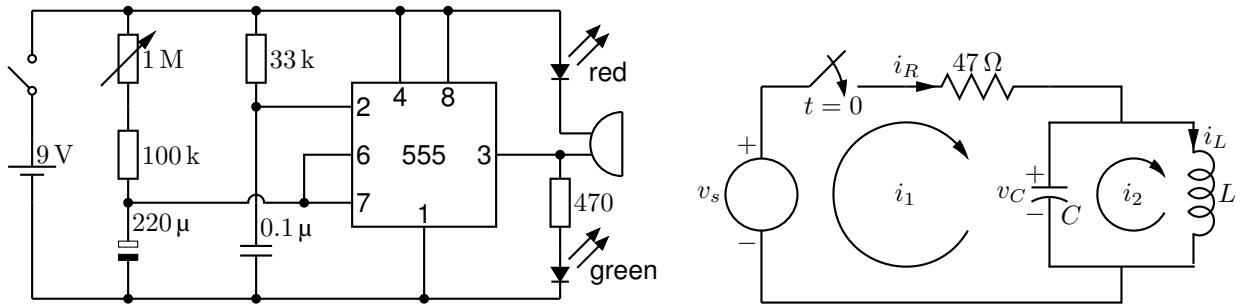
Figure 37: Two simple labeled circuits [ex01.m4].
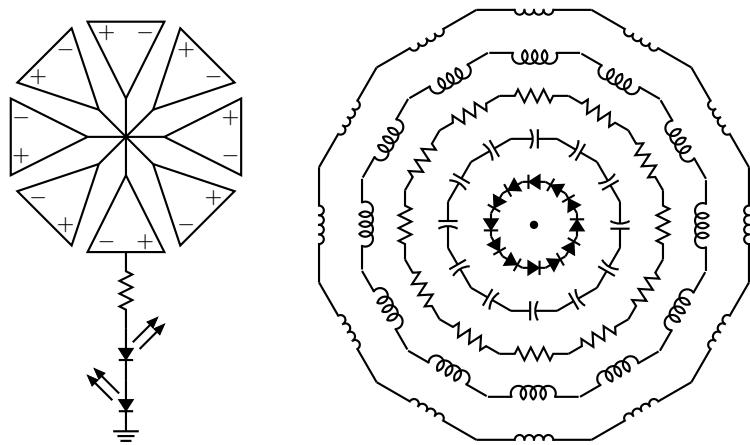
Figure 38: Elements at obtuse angles [ex02.m4].
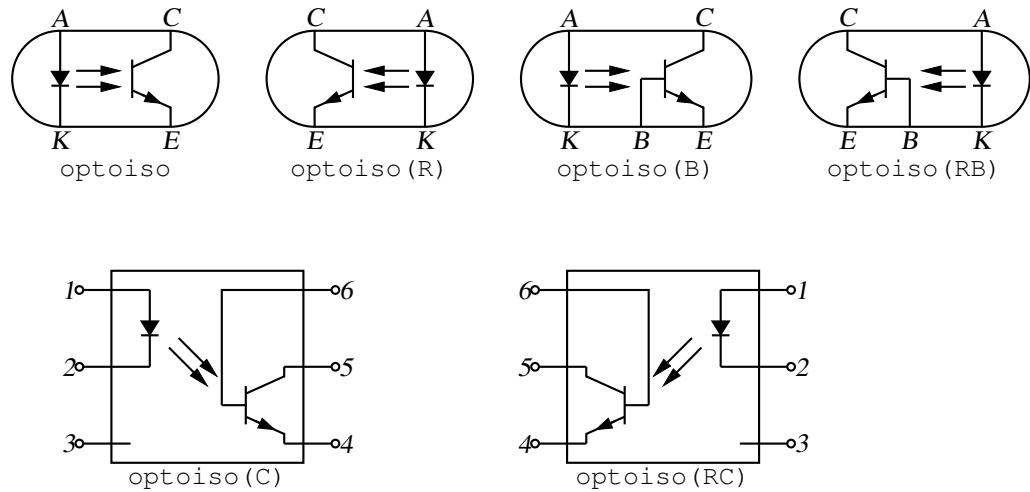
Figure 39: Optical isolator: a circuit with right or left orientation [Optoiso.m4].
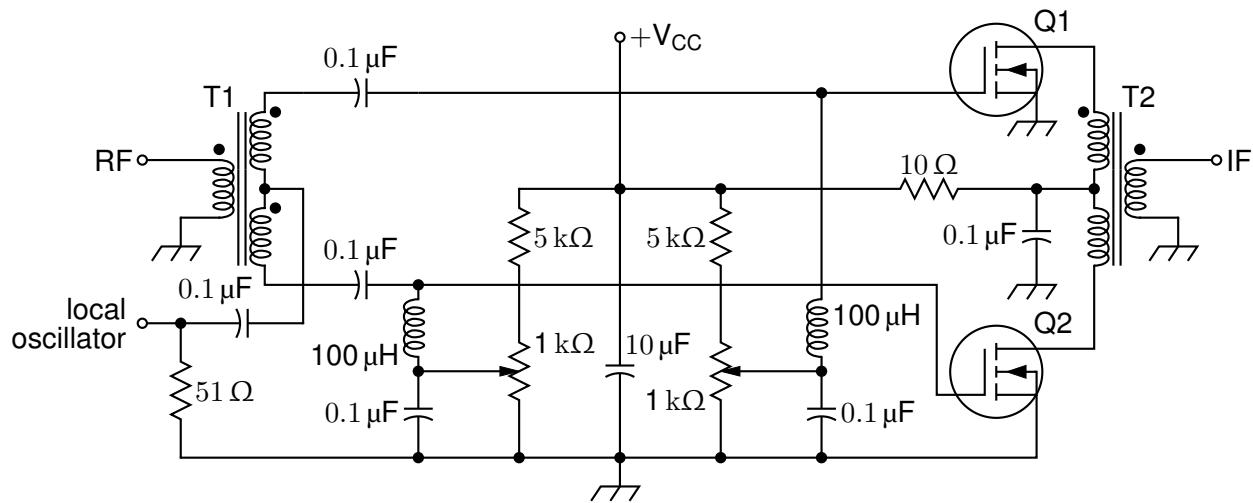
Figure 40: A balanced mixer, using `mosfet` and a custom transformer [Mixer.m4].
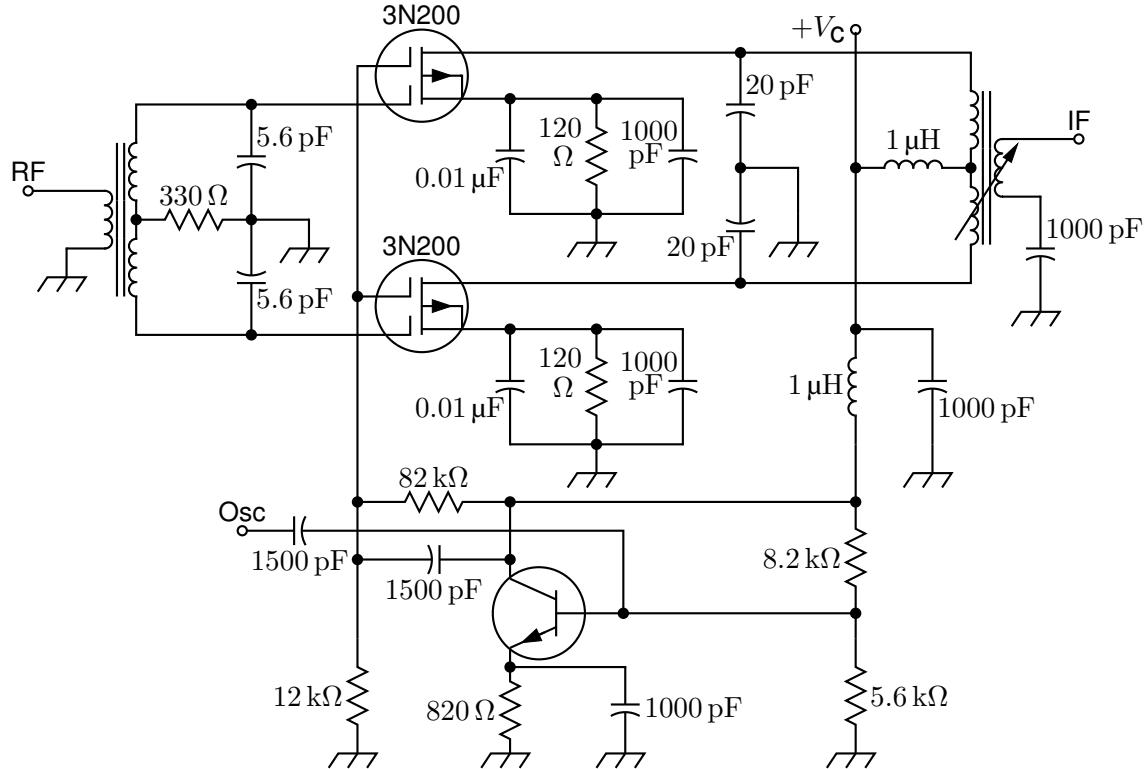
Figure 41: A push-pull mixer, showing FETs with multiple gates [PushPull.m4].
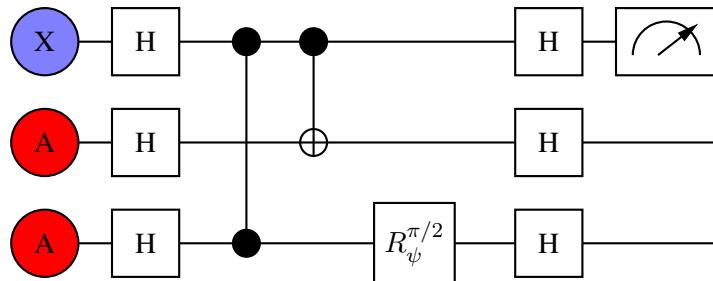
Figure 42: A quantum circuit [Quantum.m4].
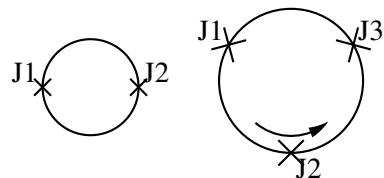
Figure 43: Superconducting quantum interface device (drawing direction down) [SQUID.m4].
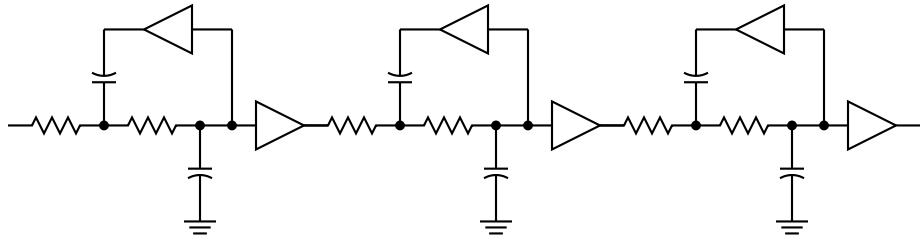
Figure 44: A six-pole filter [Sixpole.m4].
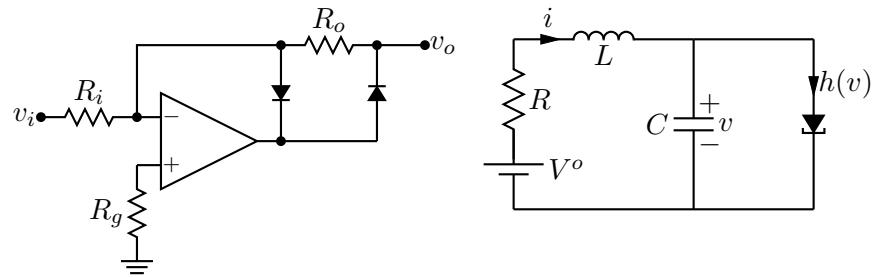
Figure 45: Precision half-wave rectifier and a tunnel diode circuit (illustrating `opamp, diode, resistor, ground,` and labels) [ex18.m4].
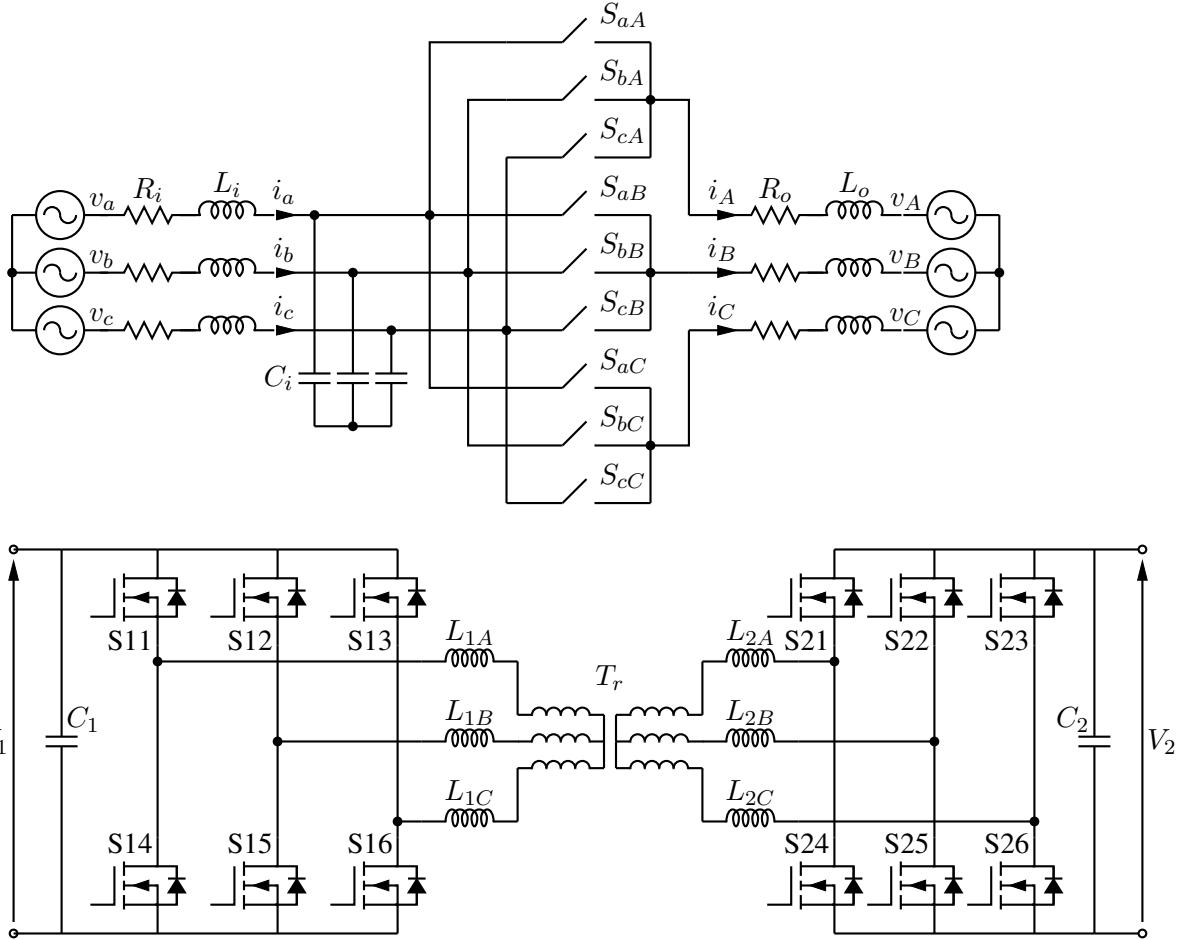
Figure 46: A three-phase switched AC-AC converter and a DC-DC converter `[MC.m4]`.
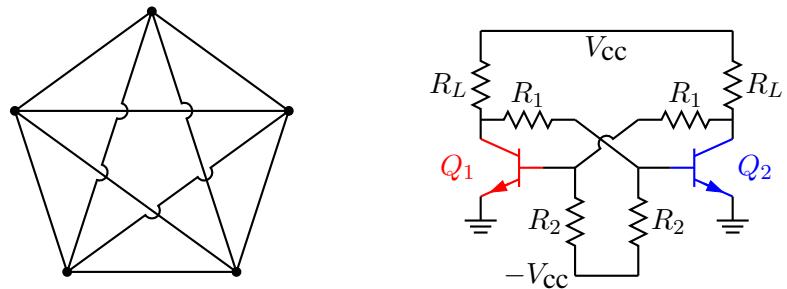
Figure 47: Non-planar graph and bistable circuit (illustrating the `crossover` macro and colored elements) `[ex10.m4]`.
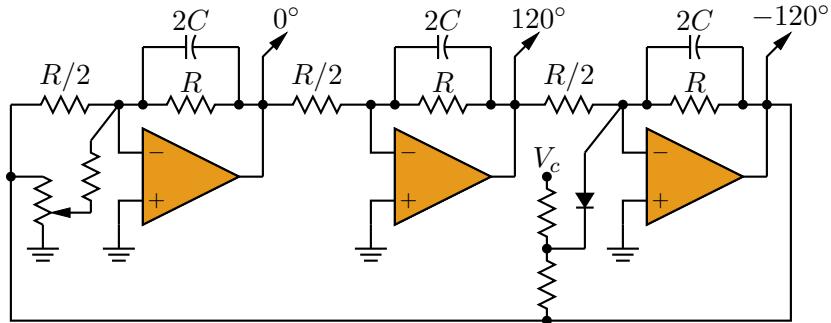
Figure 48: Three-phase oscillator `[Three.m4]`.
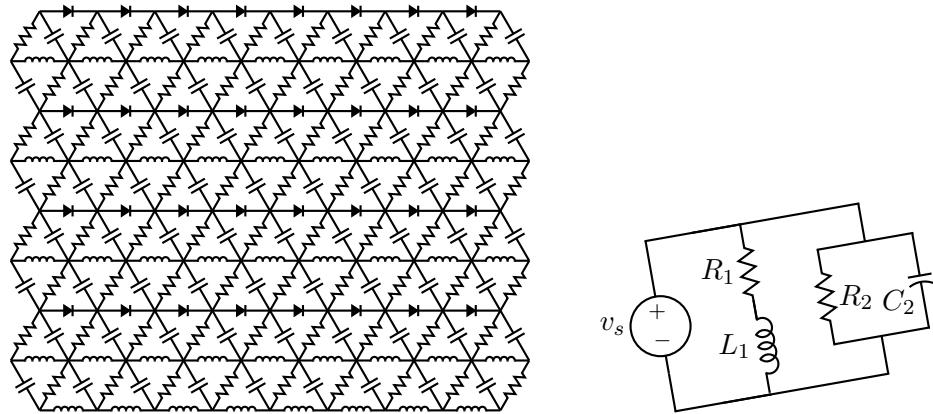
Figure 49: A repetitive network created by Pic looping and a skewed circuit used to test the macro `parallel_` `[ex17.m4]`.
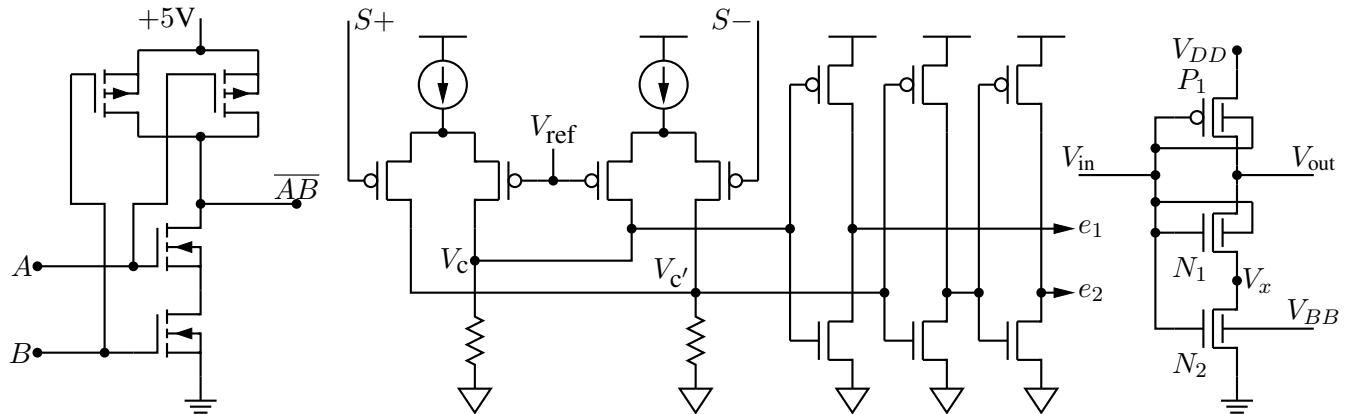
Figure 50: A CMOS NAND gate, a test circuit, and an XMOSFET example `[ex12.m4]`.
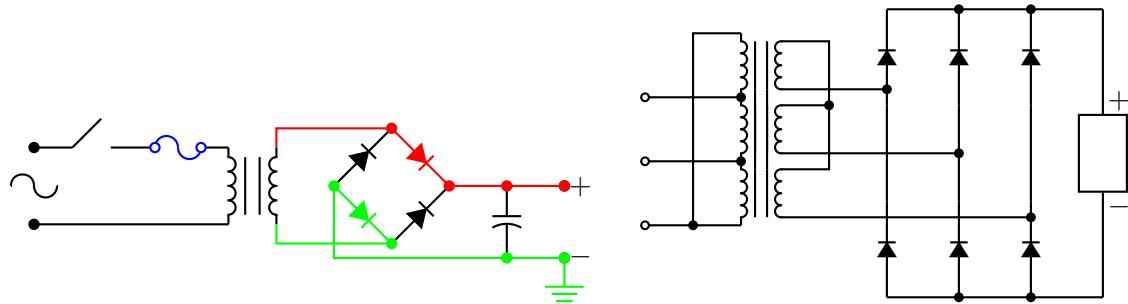
Figure 51: An elementary power supply circuit with colored elements, and a multiple-winding transformer with 3-phase rectifier [pwrsupply.m4].
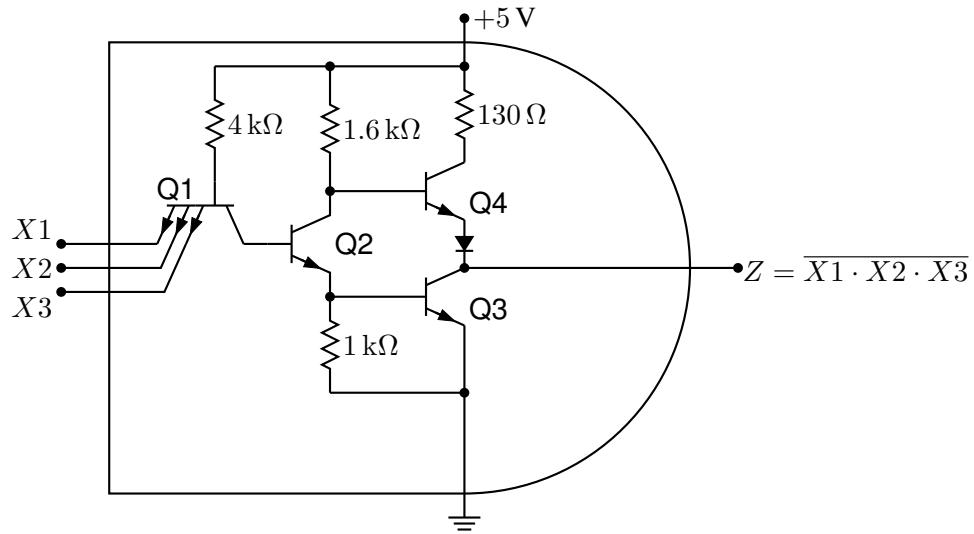

Figure 52: TTL NAND gate illustrating a transistor with multiple emitters [TTLnand.m4].
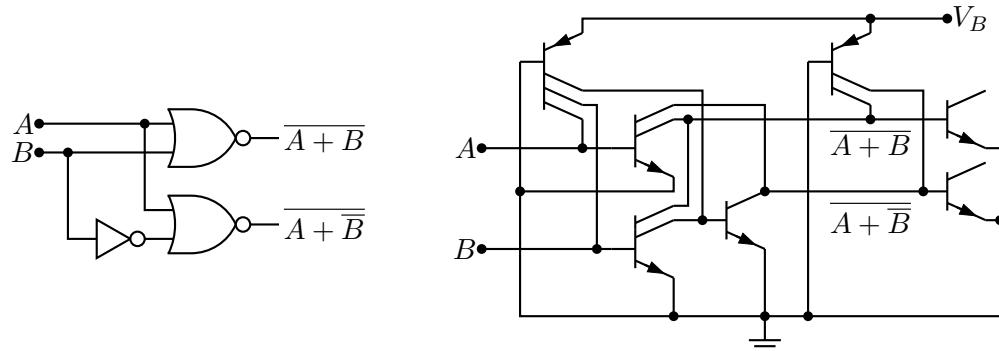
Figure 53: Gate circuit and equivalent embedded I^2L components illustrating multiple collectors [I2L.m4].
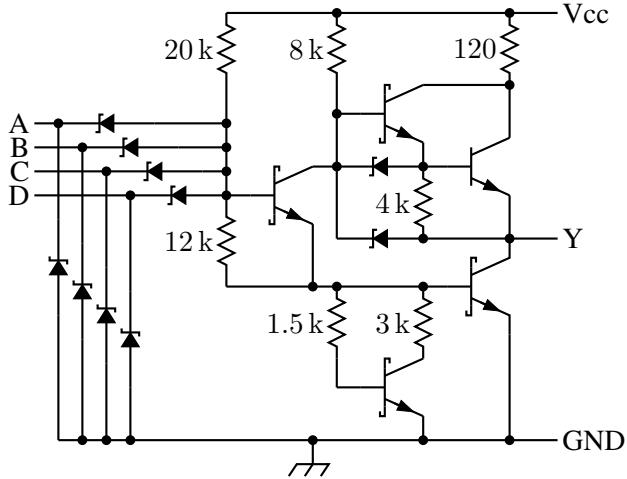
Figure 54: A 4-input NAND circuit illustrating the S (Schottky) option of bi_trans [Schottky.m4].
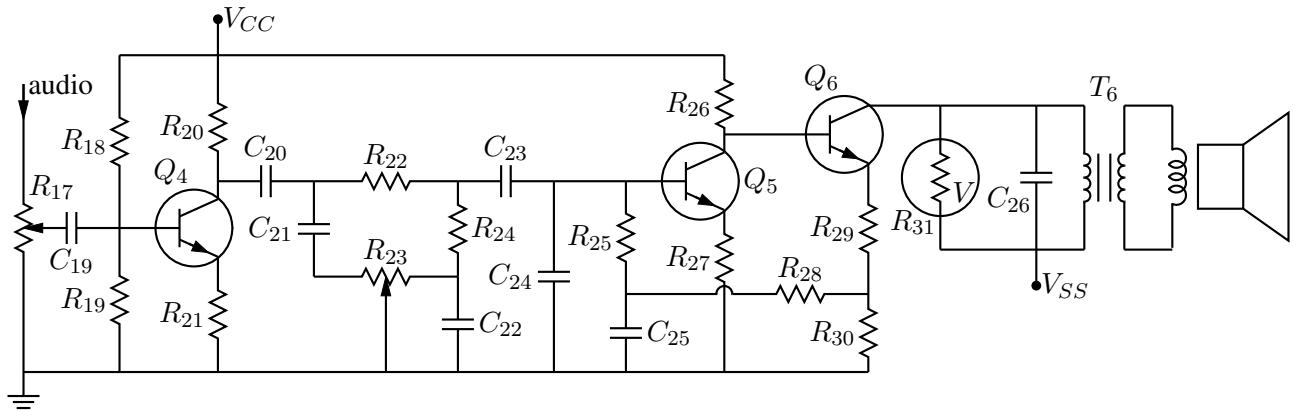

Figure 55: Transistor radio audio chain [ex11.m4].
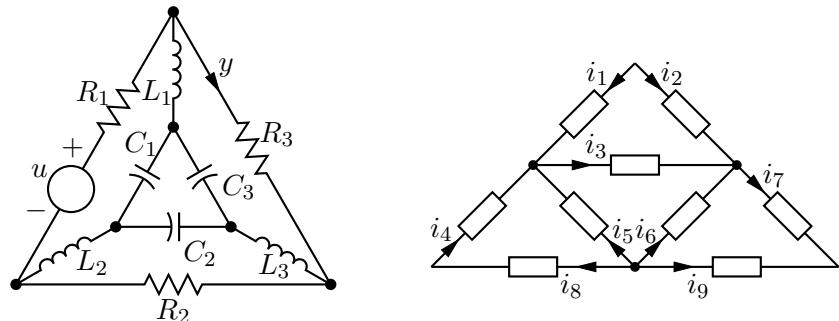

Figure 56: Labels on non-manhattan elements [ex04.m4].
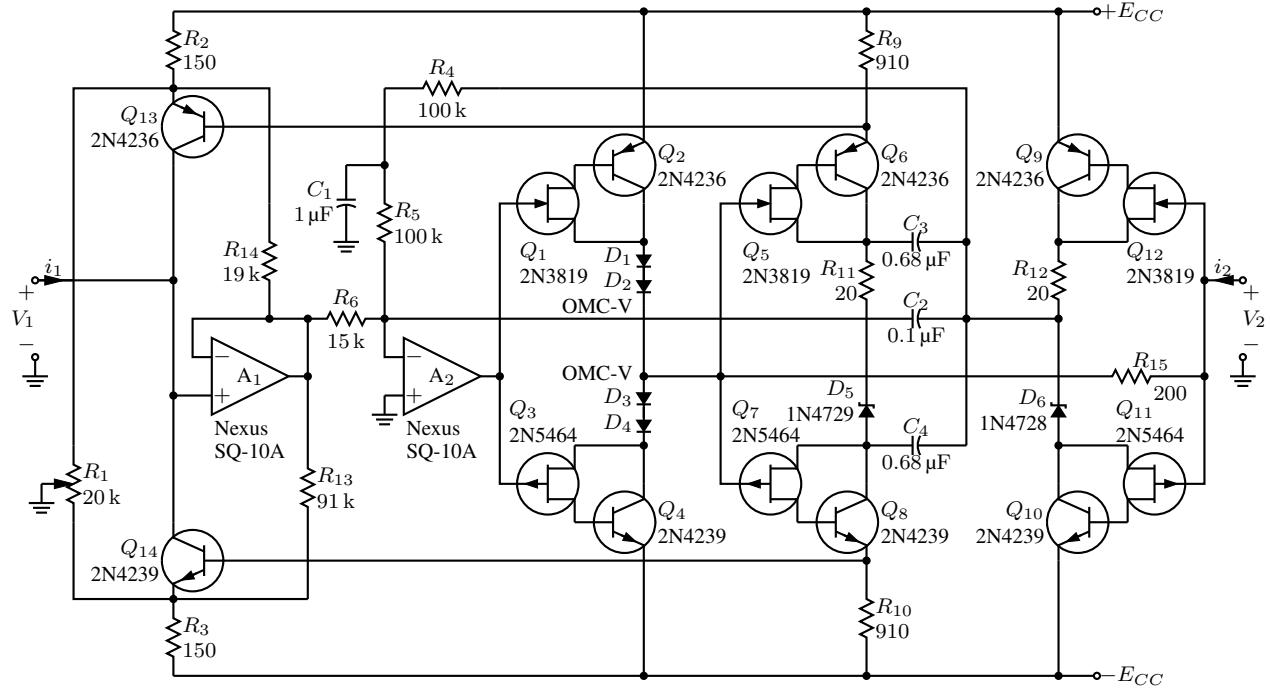
Figure 57: Realization of a controlled source (illustrating stacked element labels) `[Csource.m4]`.
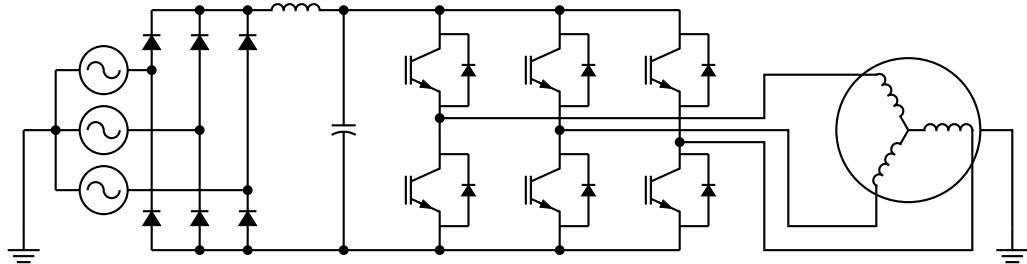

Figure 58: Synchronous machine driven by variable-speed drive and rectifier `[Drive.m4]`.
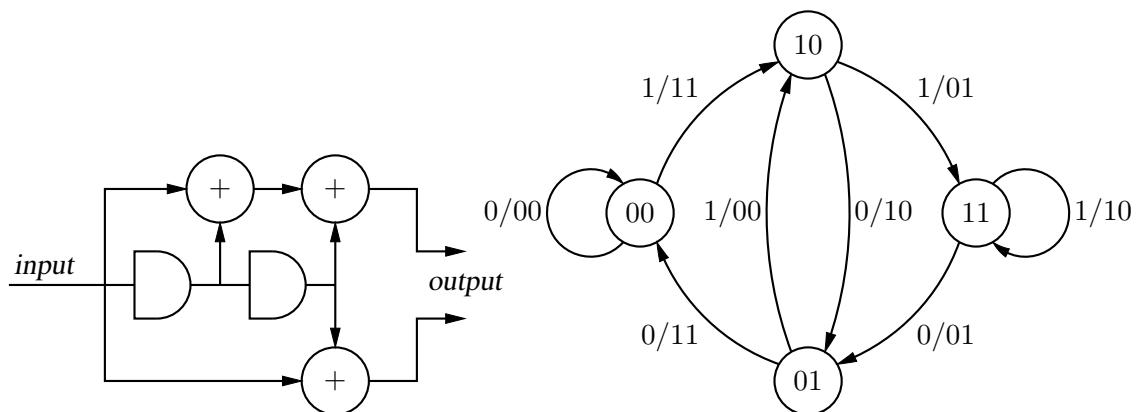
Figure 59: A rate $1/2$ binary convolutional coder and its state diagram `[ex16.m4]`.
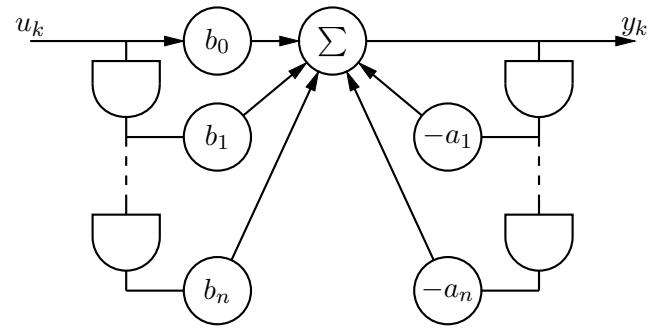
Figure 60: Digital filter [ex03.m4].
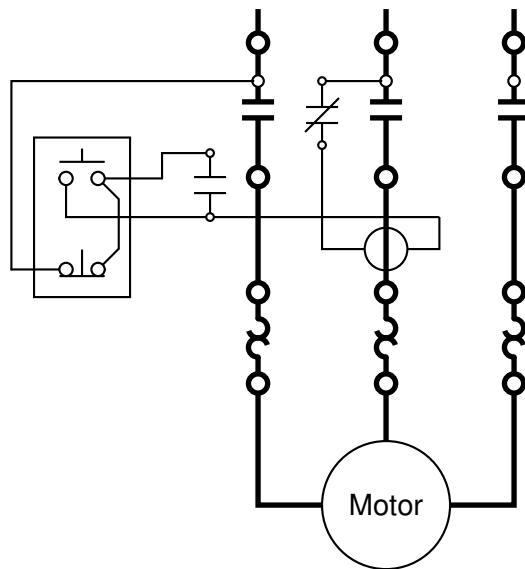
Figure 61: Motor control connections [MotorControl.m4].
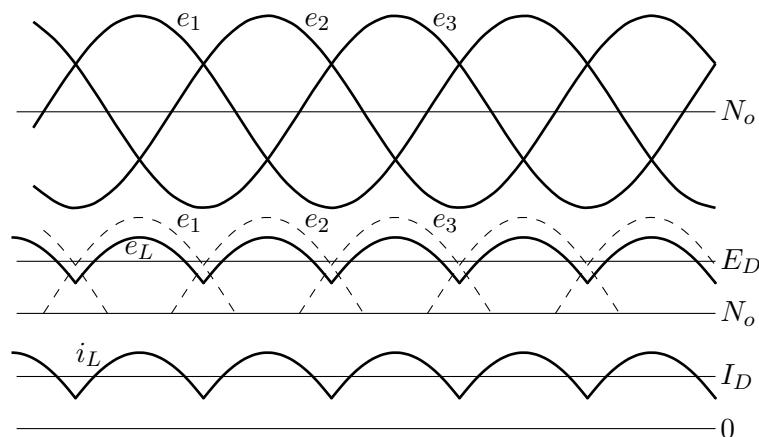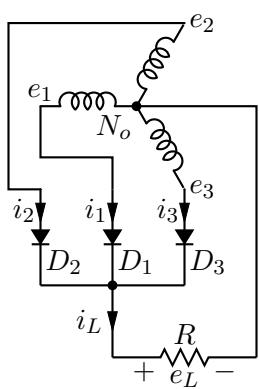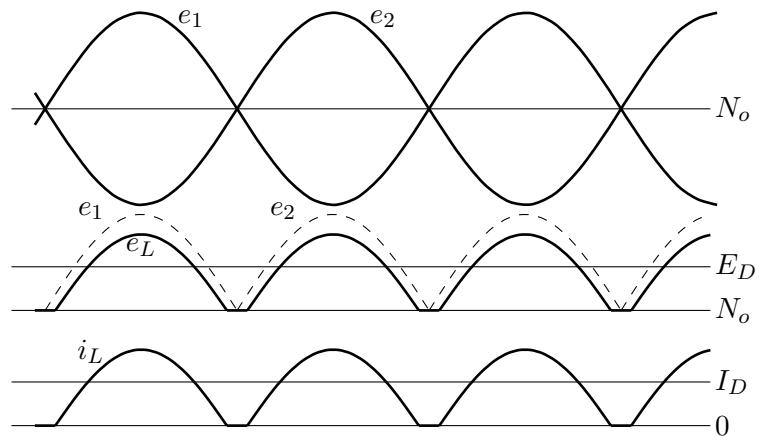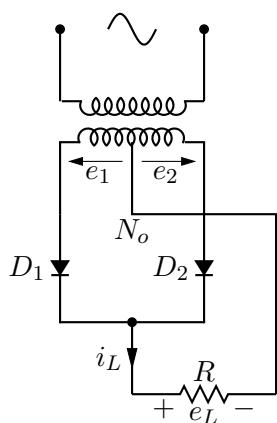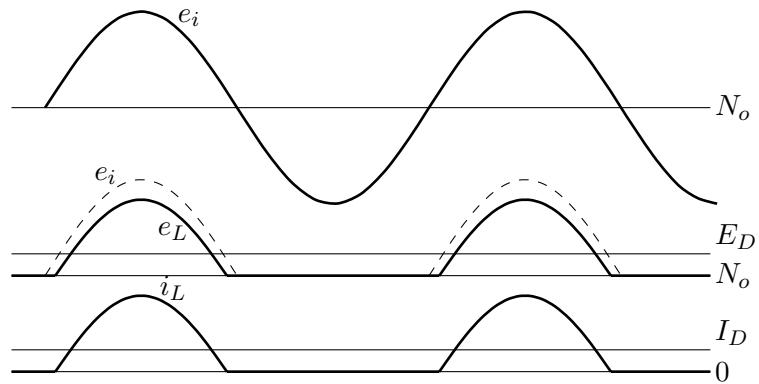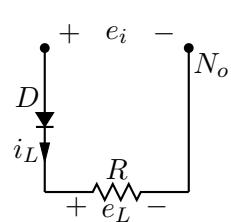
Figure 62: Rectifier circuits and waveforms [`Rectifiers.m4`].
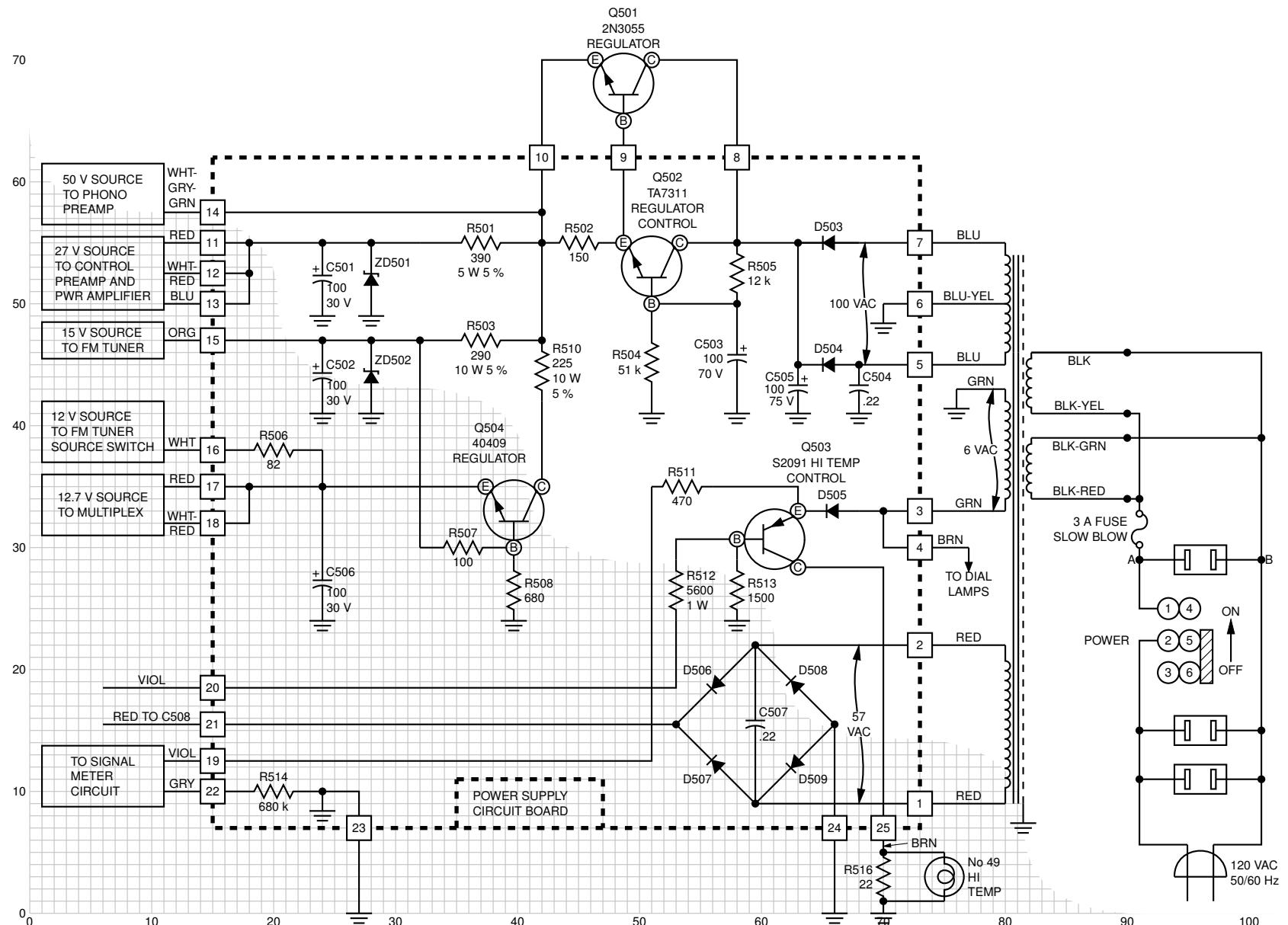
Figure 63: The power supply of a Heathkit AR-15 (Now, *that* was a receiver!) with custom transformer and other elements, drawn on a grid (partially shown) to aid in placement [Heathkit.m4].
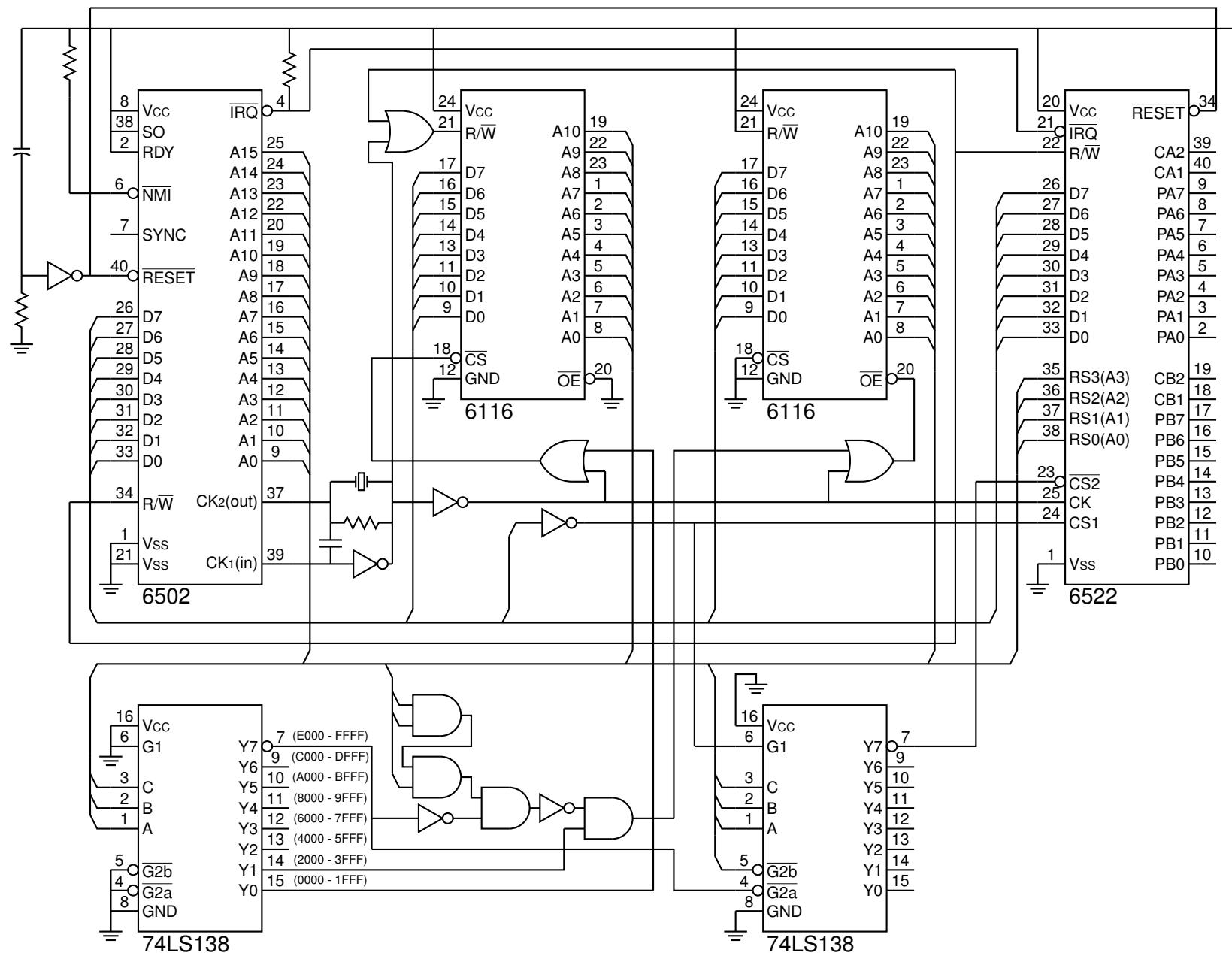
Figure 64: A digital circuit of moderate size, redrawn from M. P. Maclenan and G. M. Burns, "An Approach to Drawing Circuit Diagrams for Text Books," Tugboat (12)1, March 1991, pp. 66-69 [lcct.m4].
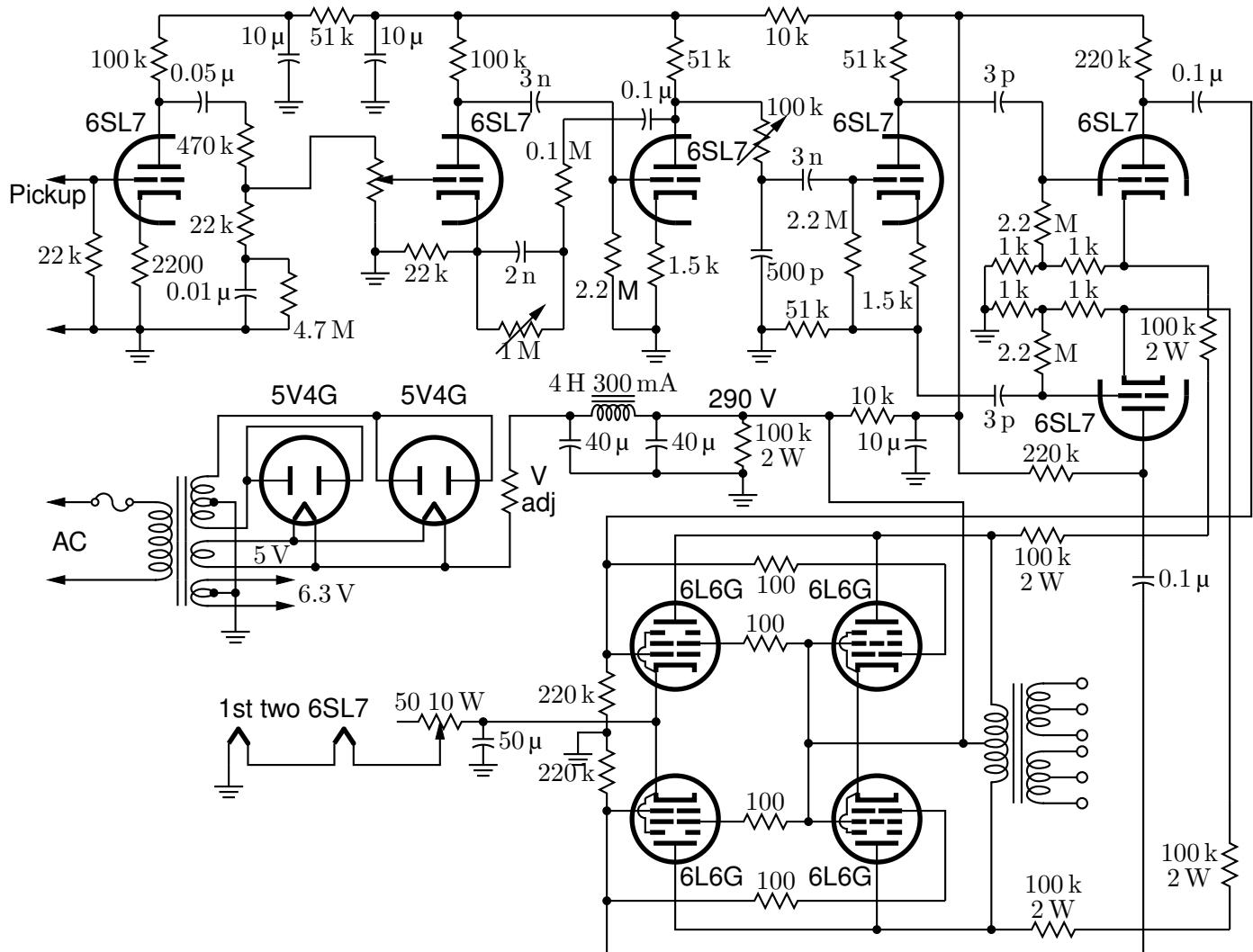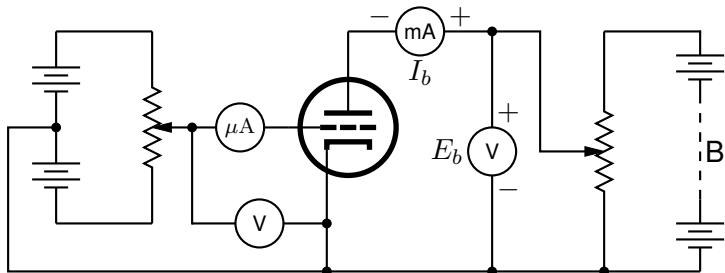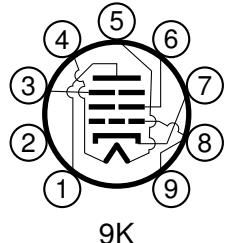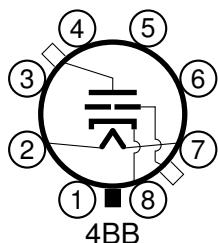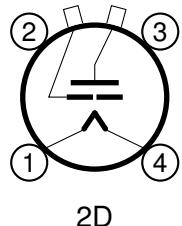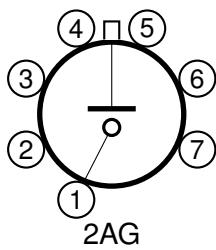
Figure 65: Electron-tube diagrams: a few bottom-view base diagrams, a generic triode test circuit, and a 25-watt audio amplifier adapted from F. Langford-Smith, *Radiotron Designer's Handbook,* fourth edition, Harrison, NJ: Radio Corporation of America, 1952 [Tubediags.m4].
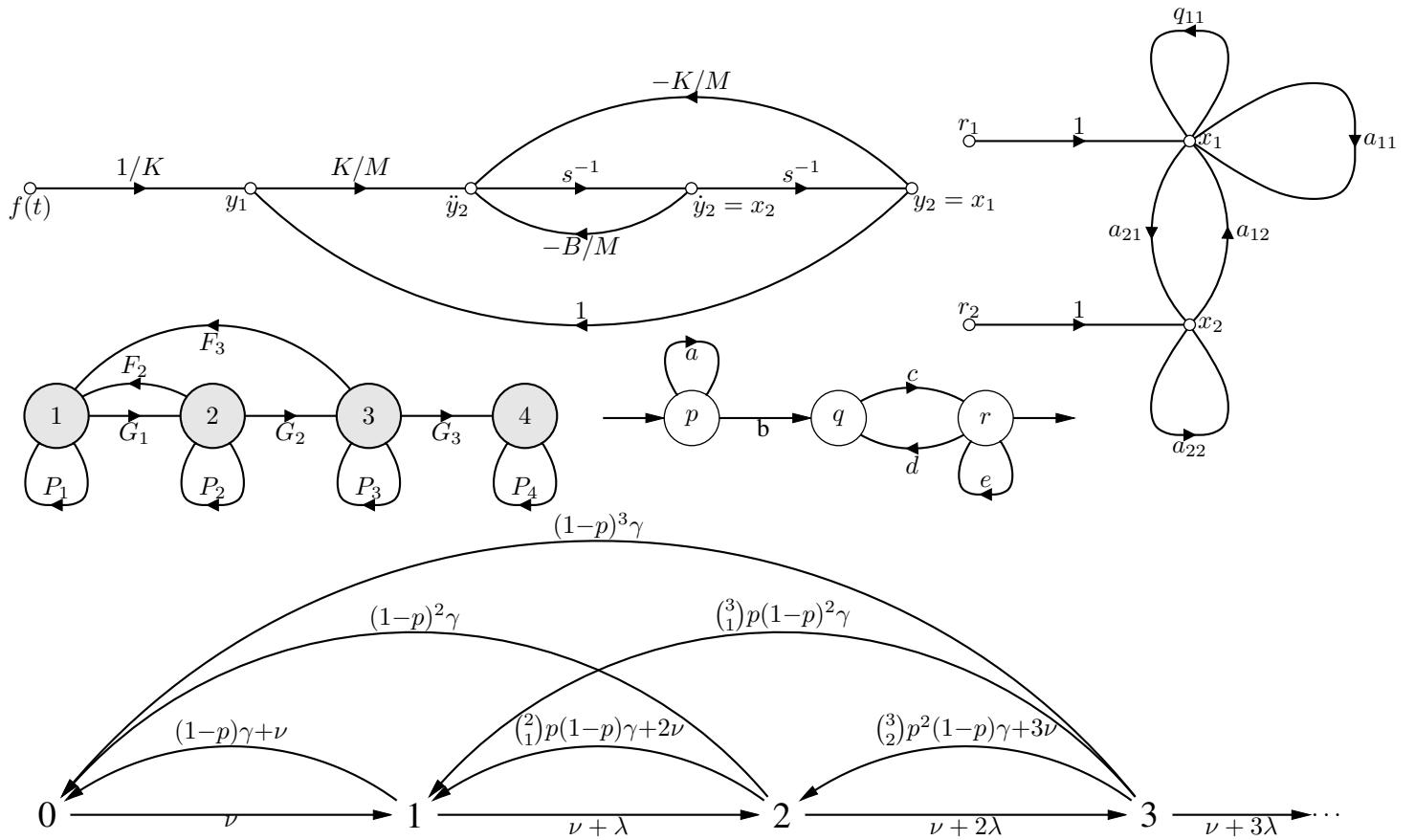
Figure 66: Signal-flow graphs [sfg.m4].
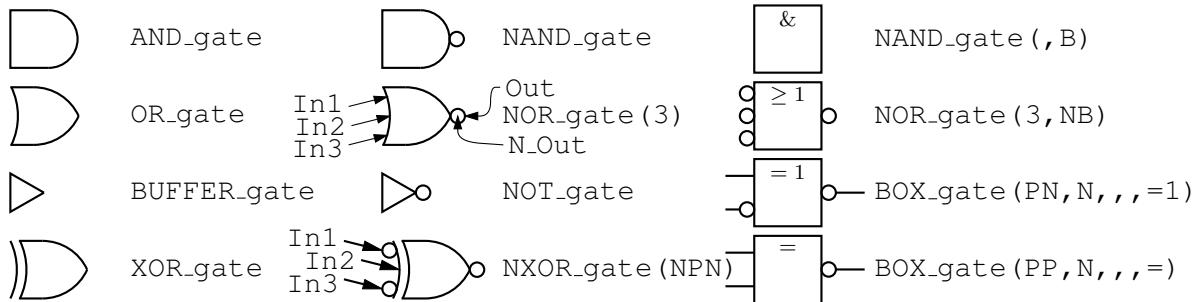
Figure 67: Basic logic gates [Logic.m4].
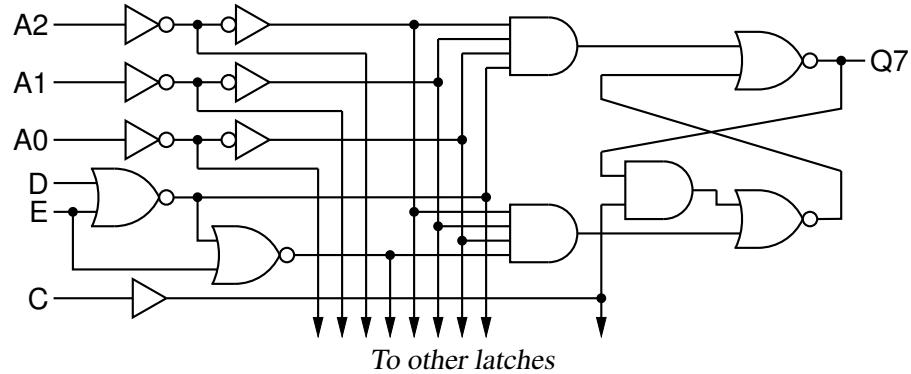
Figure 68: General-purpose latch: a small logic circuit [ex08.m4].
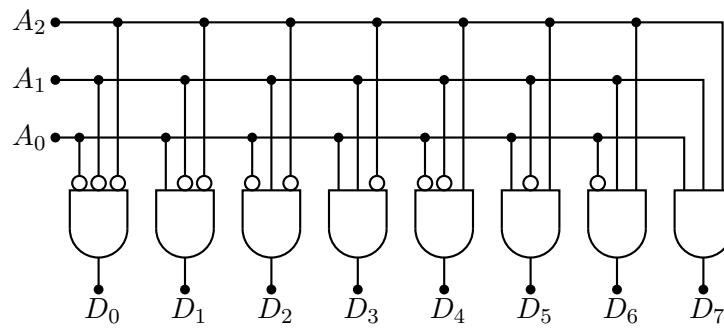
Figure 69: Decoder logic, constructed using the for_ macro [Decoder.m4].
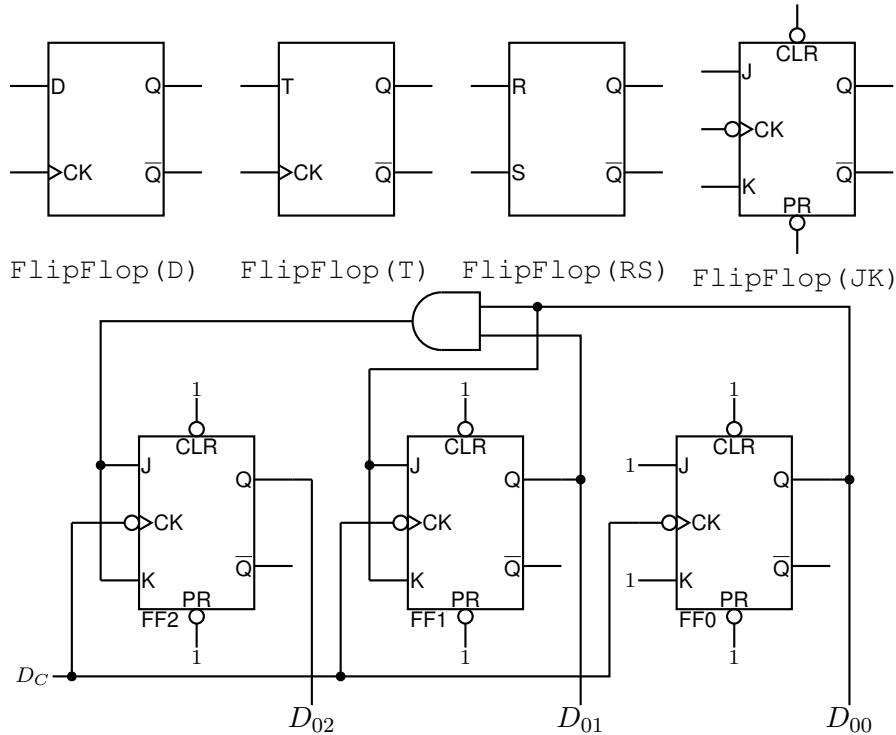

Figure 70: Some flip-flops [ex21.m4].
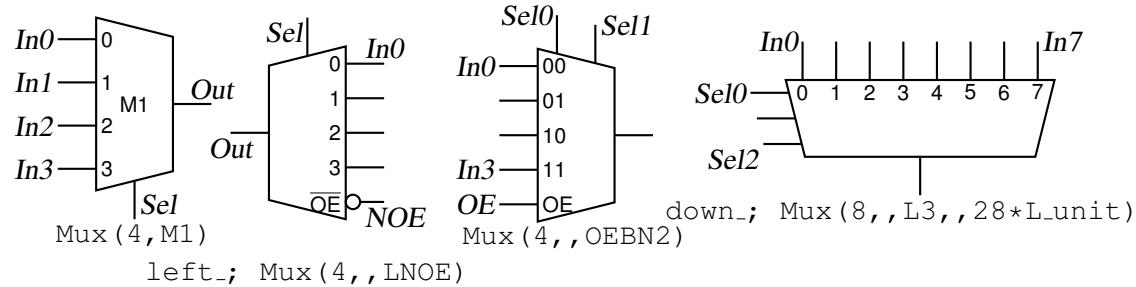
Figure 71: Multiplexer [Multiplexer.m4].
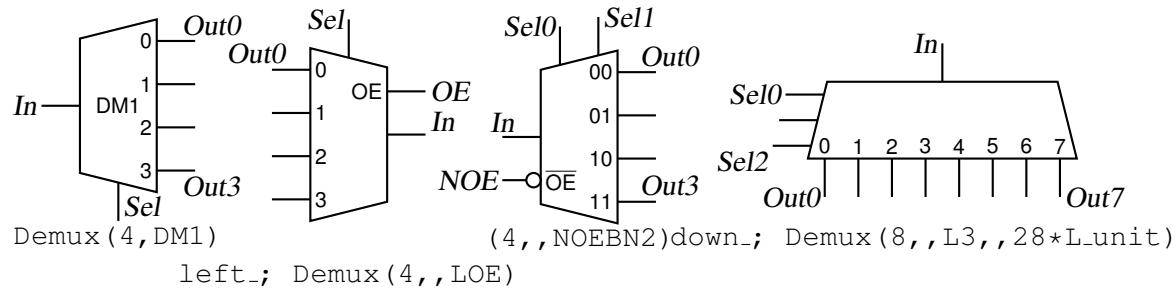
Figure 72: Demultiplexer [Demultiplexer.m4].
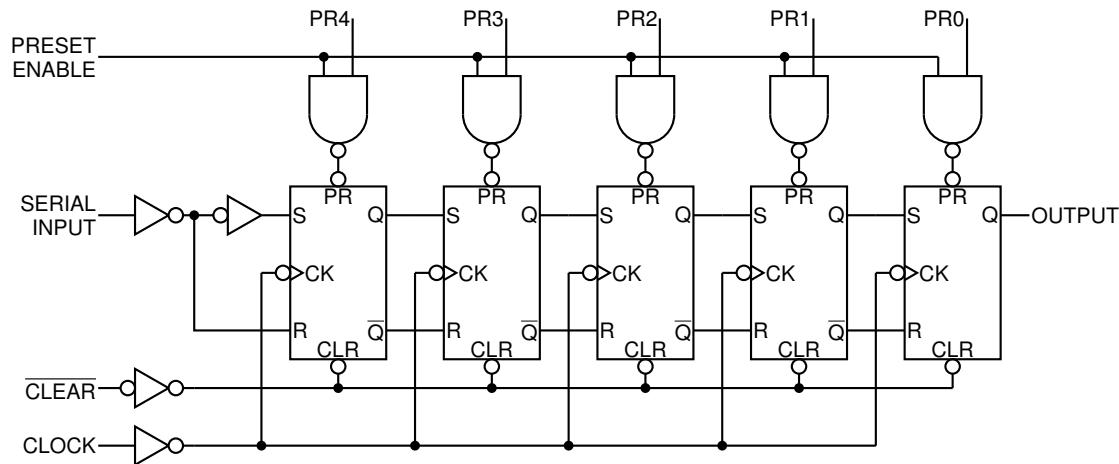
Figure 73: A 5-bit shift register drawn using a custom flip-flop [ShiftR.m4].

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
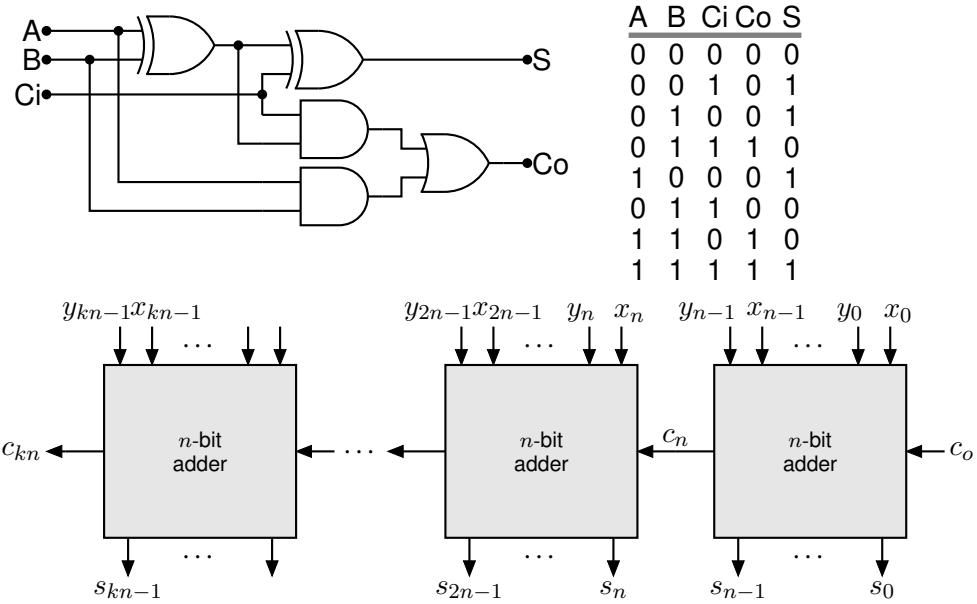
Figure 74: A full adder and a cascade of n-bit adders [Adder.m4].

CanLogic(AND,,OR,,abcd,~ba,c,d~a)   CanLogic(OR,N,NAND,,ab~c,a~bc,ac,~d)
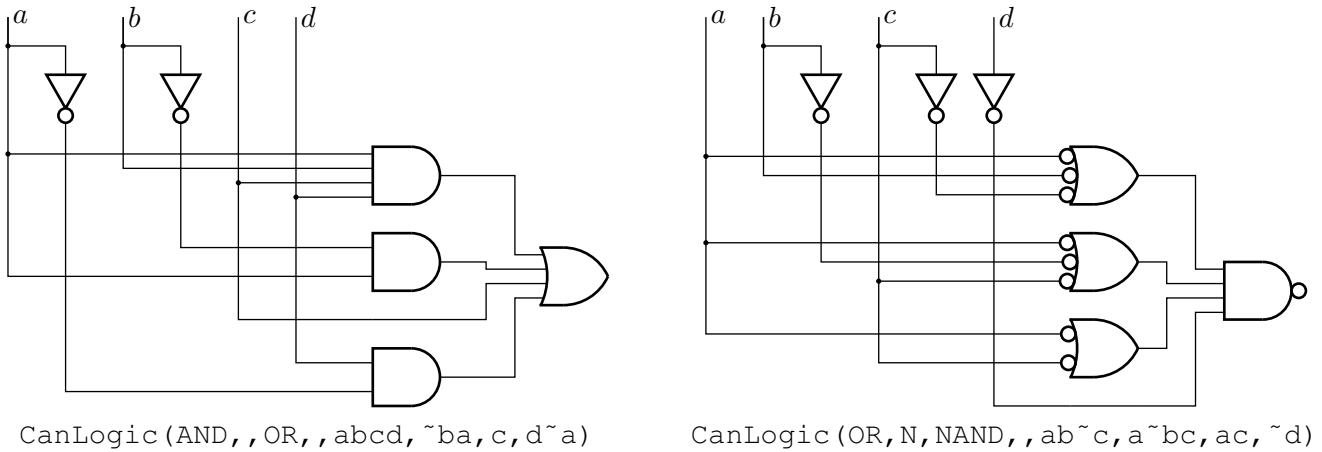
Figure 75: A way of automatically drawing two-layer logic diagrams [CanLogic.m4].

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

```
Out1
Out2
```

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

```
B: Autologix(
  Or(And(B,Not(C)),And(Not(A),B,C)),N)
```
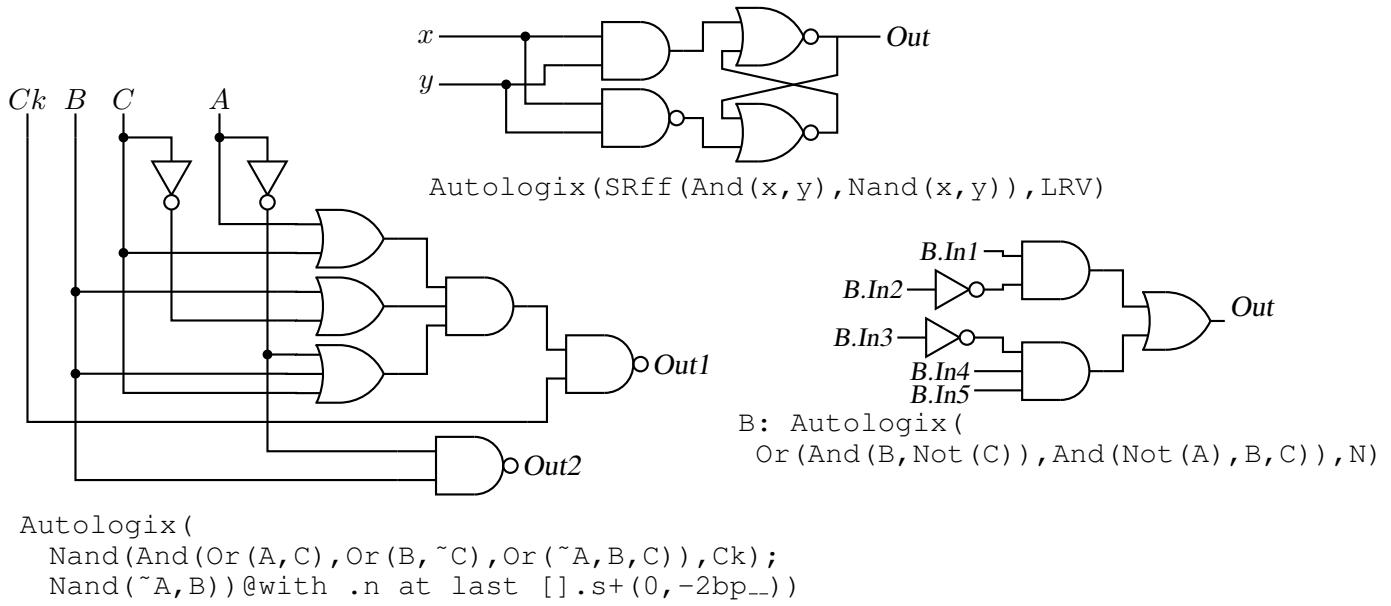
Figure 76: The `Autologix`(*Boolean expression*; *Boolean expression...* , *options*) macro automatically draws Boolean expressions in function notation. The function tree is drawn, then a row or column of inputs, then the connections. The default result is on the left, a custom element at the top, and a tree of gates only is shown on the right. THe lower examples illustrate the L, R, and V options.) `[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)
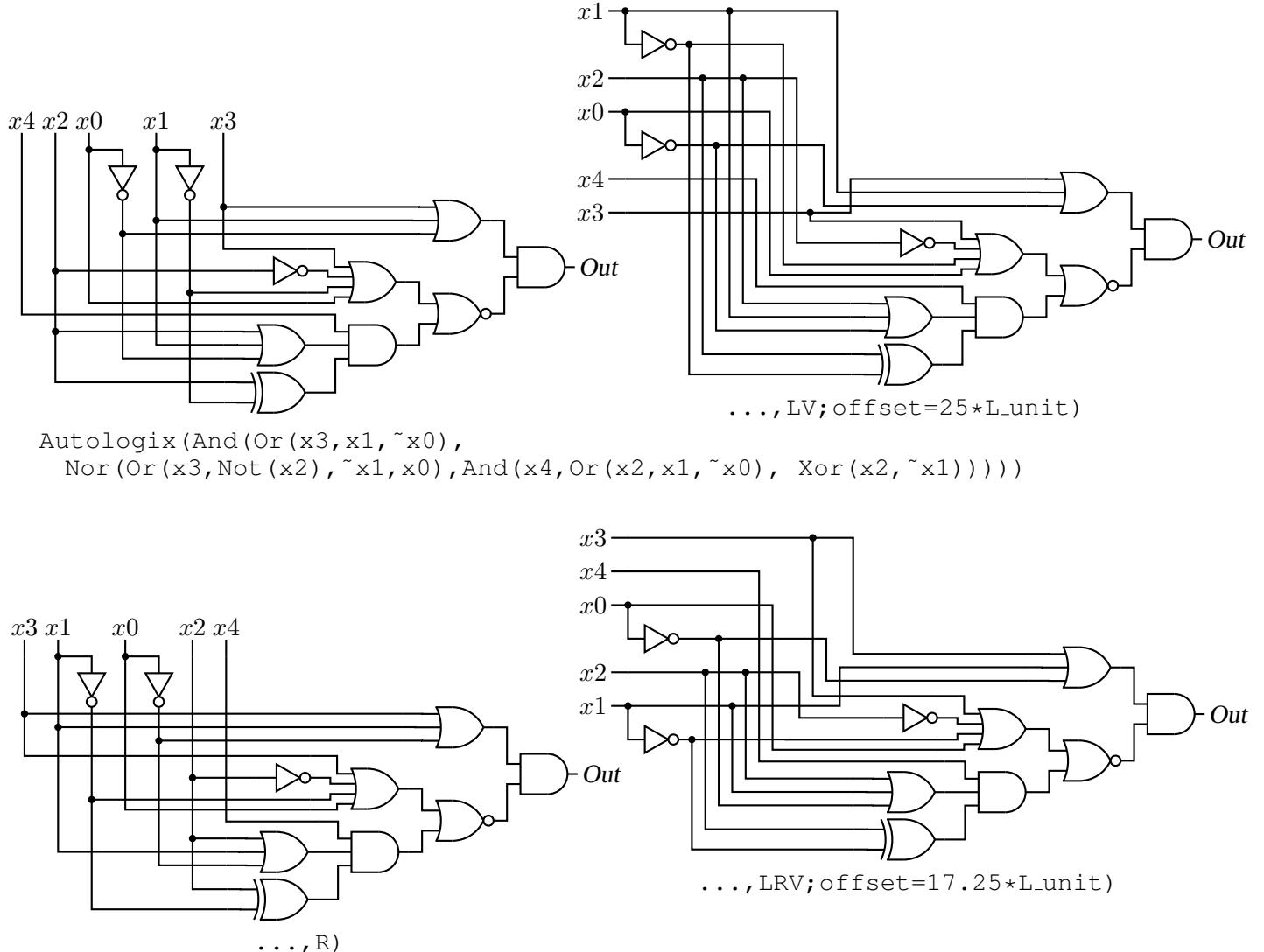
Figure 77: The Autologix macro can draw inputs on the left but the added drawing complexity may require hand tuning with second-argument options: L puts the inputs on the left, R reverses their order, V scans the input arguments in reverse order, and offset=*value* displaces the array of inputs [ABlogix.m4].
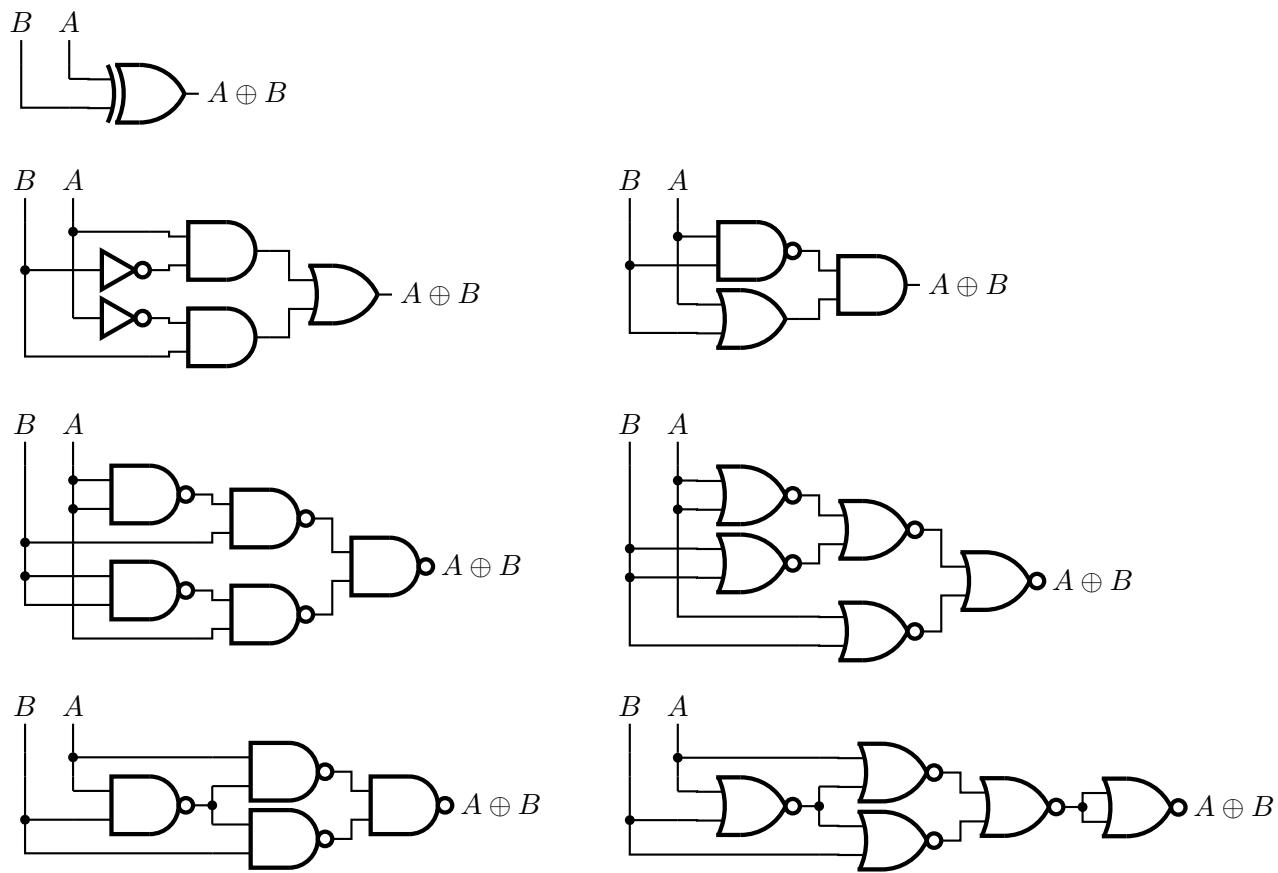
Figure 78: Realizations of the XOR function using `Autologix` [XOR.m4].
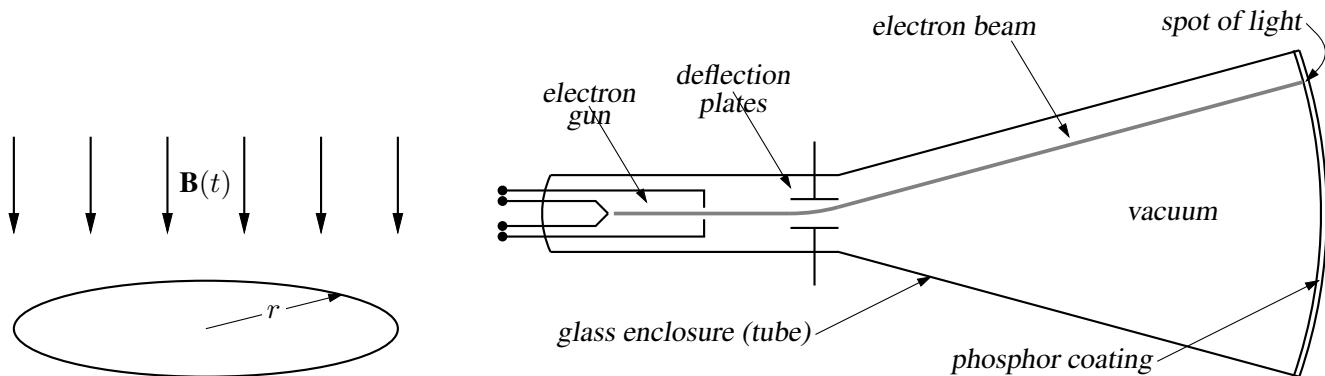


Figure 79: Line diagrams [ex00.m4].
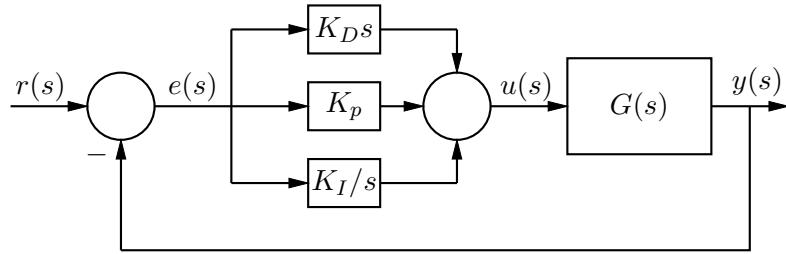
(a) PID control
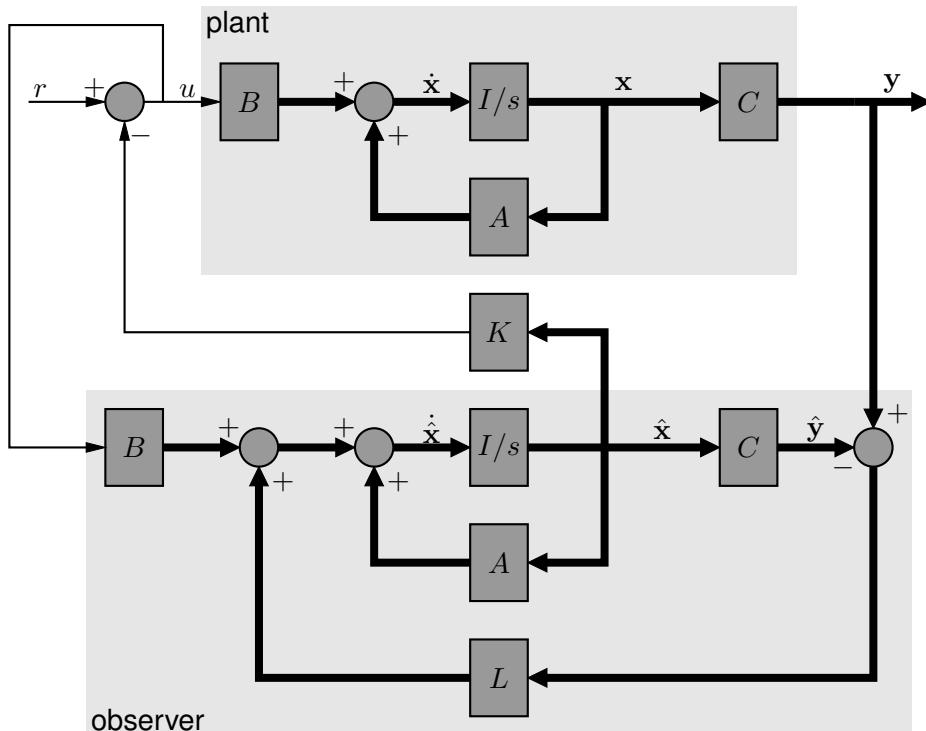
(b) Single-input plant with feedback from a full-order observer
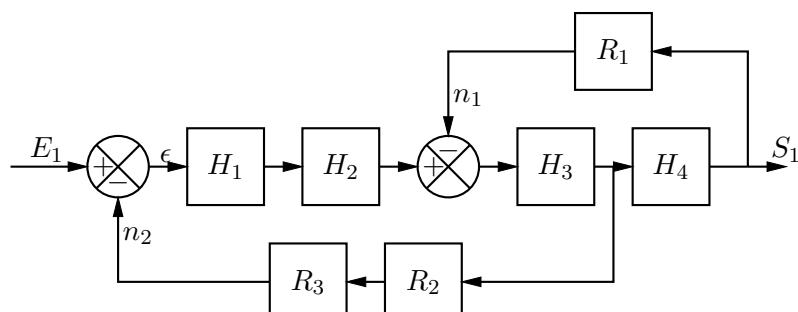
(c) A multiblock example
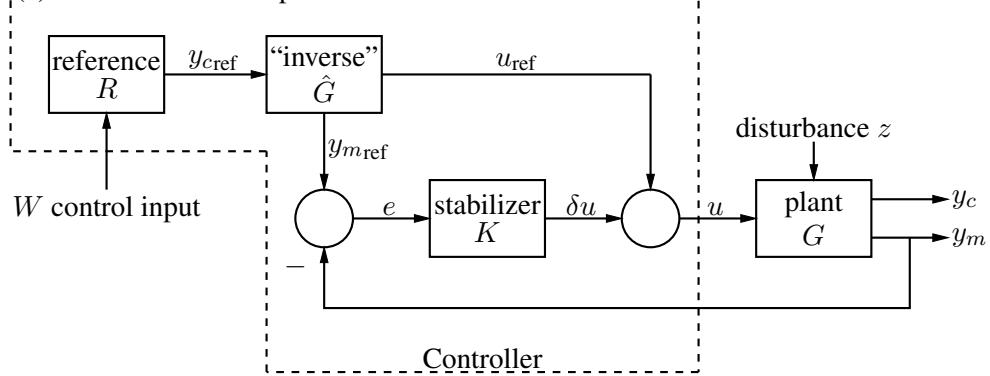
(d) Nonlinear feedforward (for performance) and small-signal feedback (for stability)

32

Figure 80: Control-system block diagrams [control.m4].
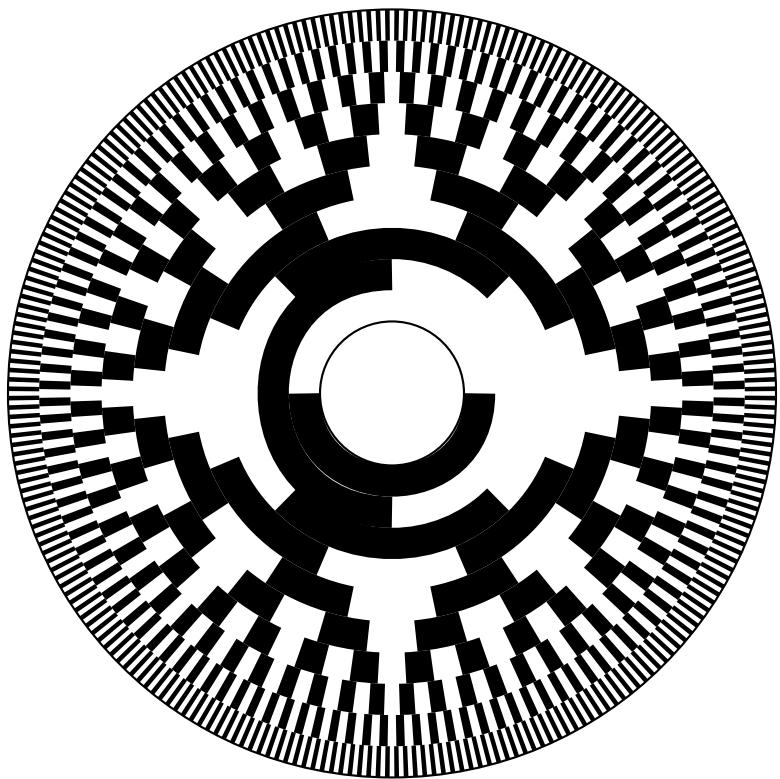
Figure 81: Gray code 10-bit encoder disk pattern [GrayCode.m4].
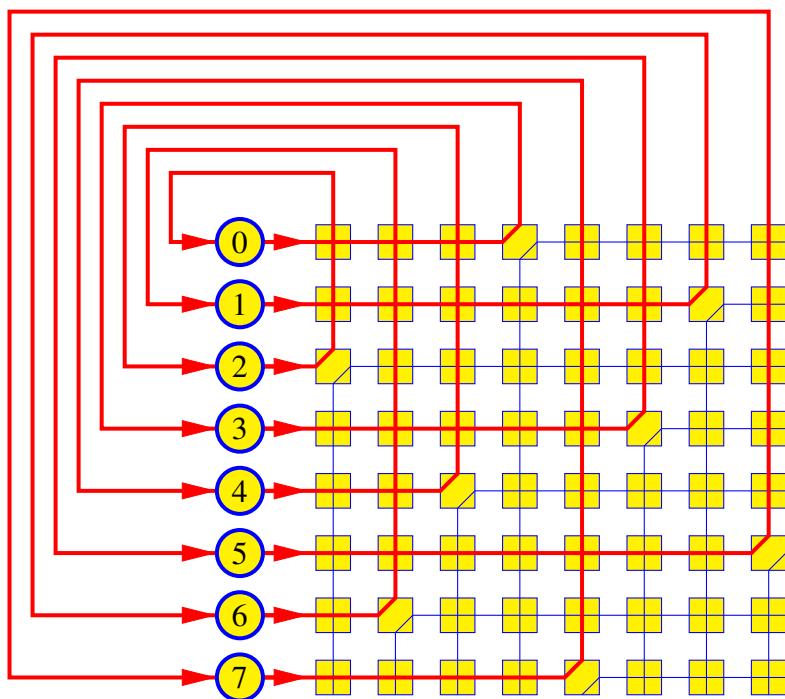
Figure 82: A crossbar switch [Crossbar.m4].

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)
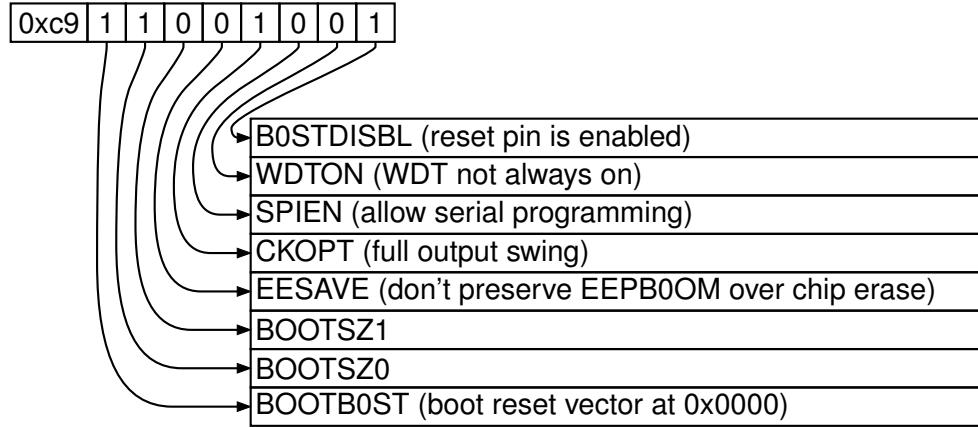
Figure 83: Elementary splines [Byte.m4].

r=0.1

t=0.2   t=0.4   t=0.6   t=0.8

t=1   t=1.2   t=1.4   t=1.6
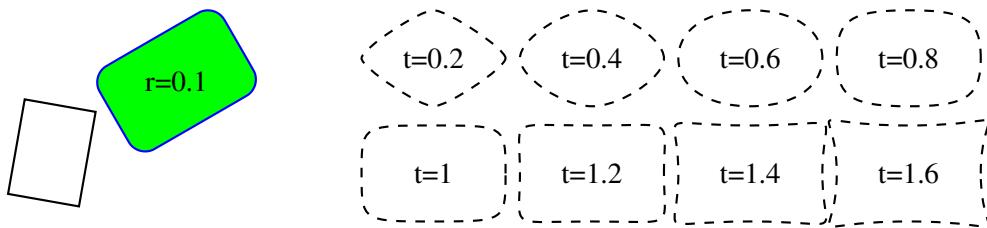
Figure 84: The macro rotbox(*wid,ht,type,*[r|t=*val*]) draws a box in the current direction [Rotbox.m4].

k t k t $n+p+m$

0   0   0   0

$\}r$

ℓ

q

F E G
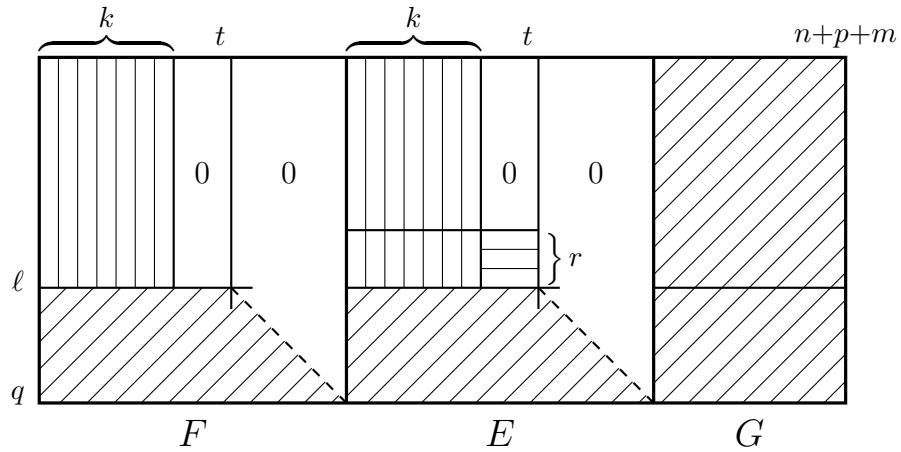
Figure 85: Crosshatching by for loops [ex06.m4].

2.36 4.22 (2) PL

10.92

27.76

13.28

30.15

41.28

A AB B

CA X DB

C D

T^{15}

C to D

blank width

label

offset

tic offset

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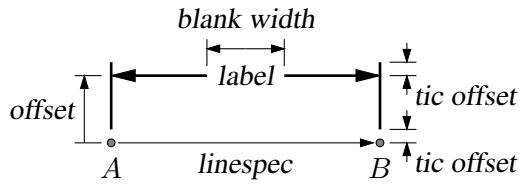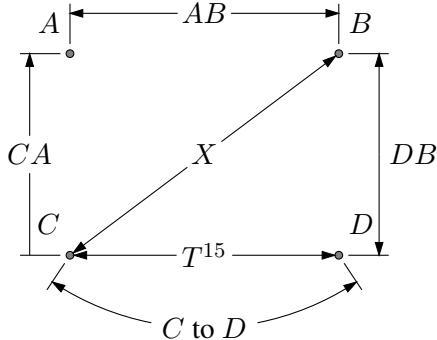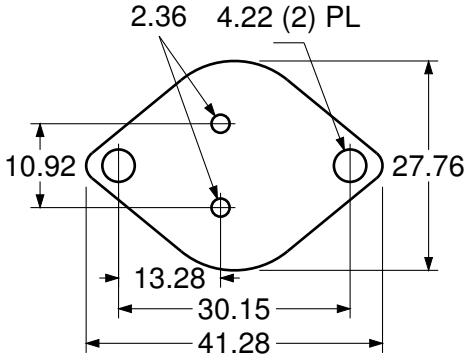
```
dimension_(from A to B,0.3,$AB$,20bp__)
dimension_(from C to B,,$X$,16bp__)
dimension_(from C to A,0.3,$CA$,14bp__,,->)
dimension_(from D to B,-0.3,"$DB$" ljust)
dimension_(from C to D,,s_box($T^{%g}$,15),W)
arcdimension_(from C to D with .c at \
    0.5 between A and B,12bp__,s_box($C$ to $D$),W)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

Figure 86: 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].
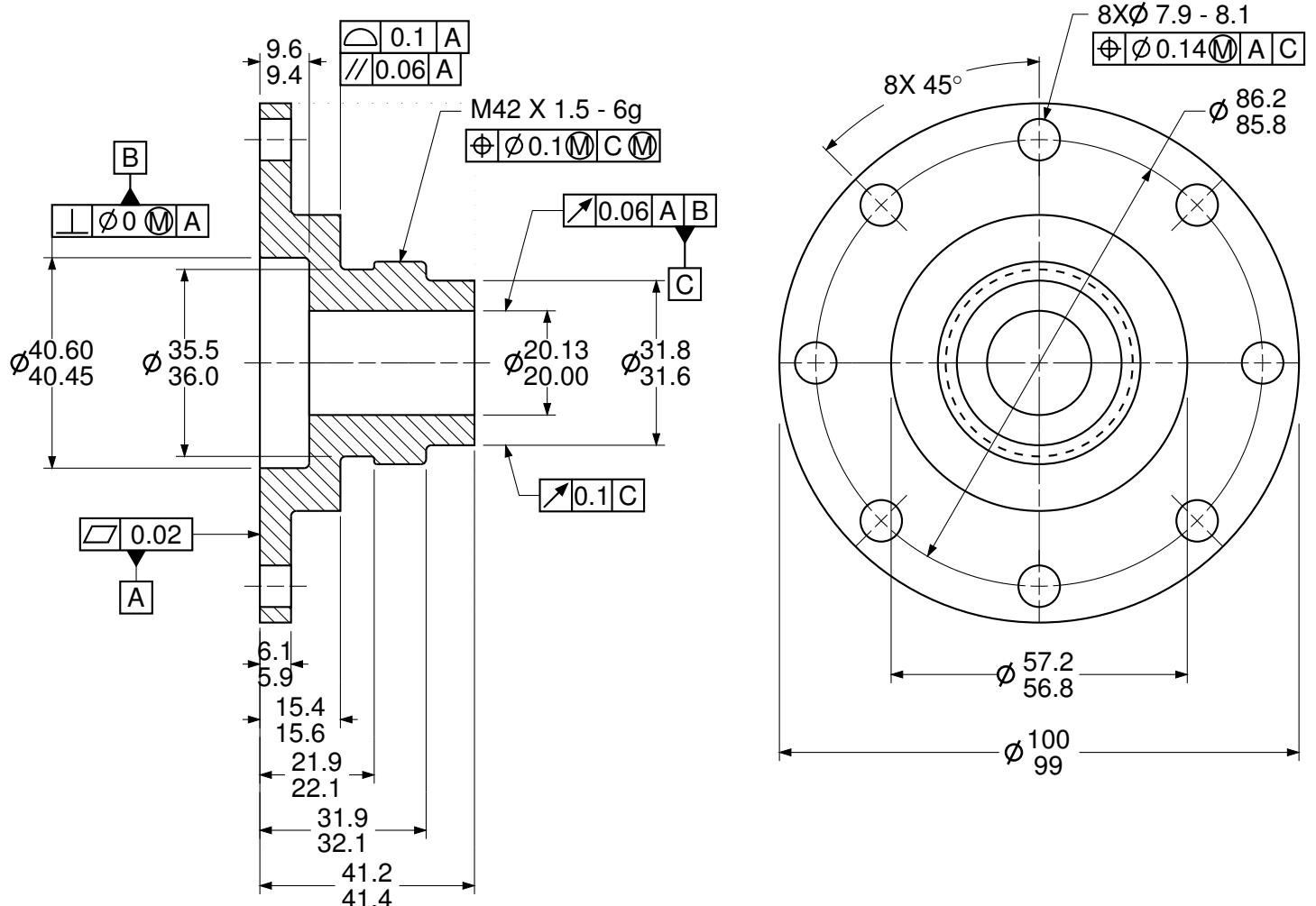
Figure 87: Dimensioning with tolerances [`Plate.m4`].
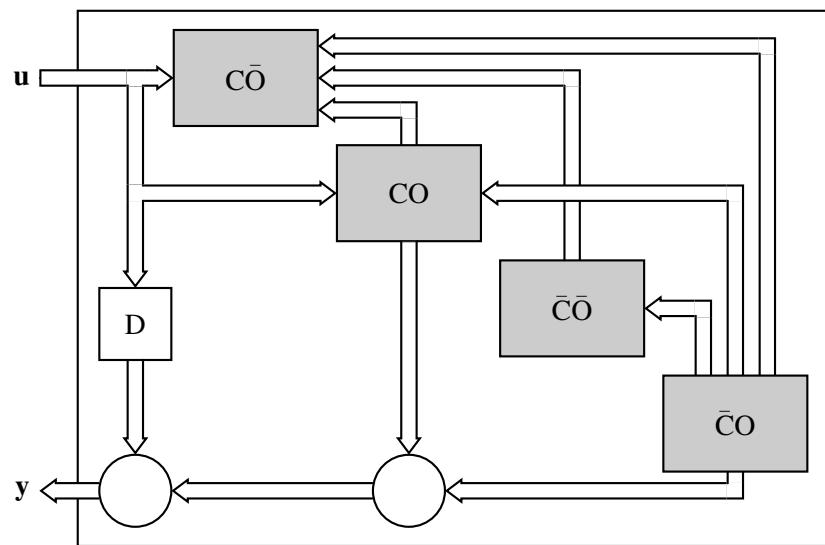

Figure 88: Use of `darrow` [`ex05.m4`].
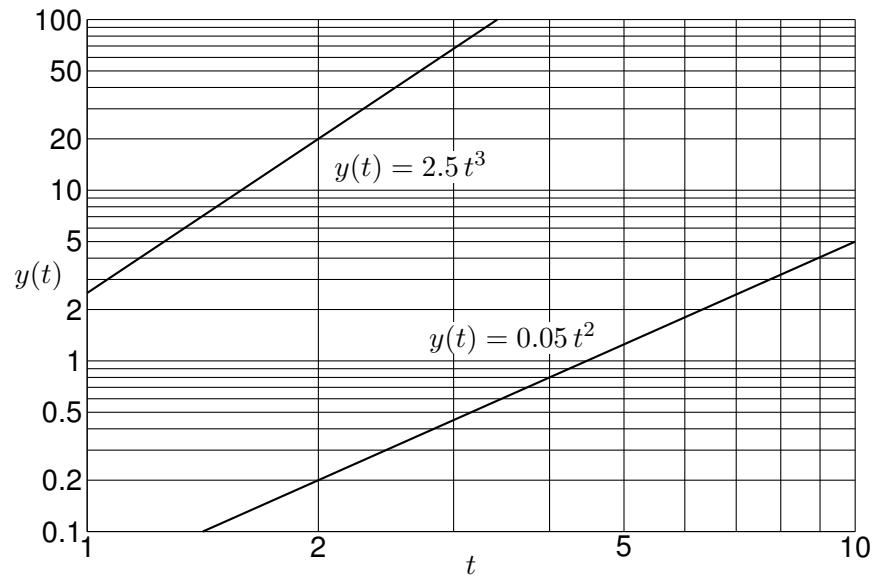
Figure 89: A graph drawn using the pic language [Loglog.m4].
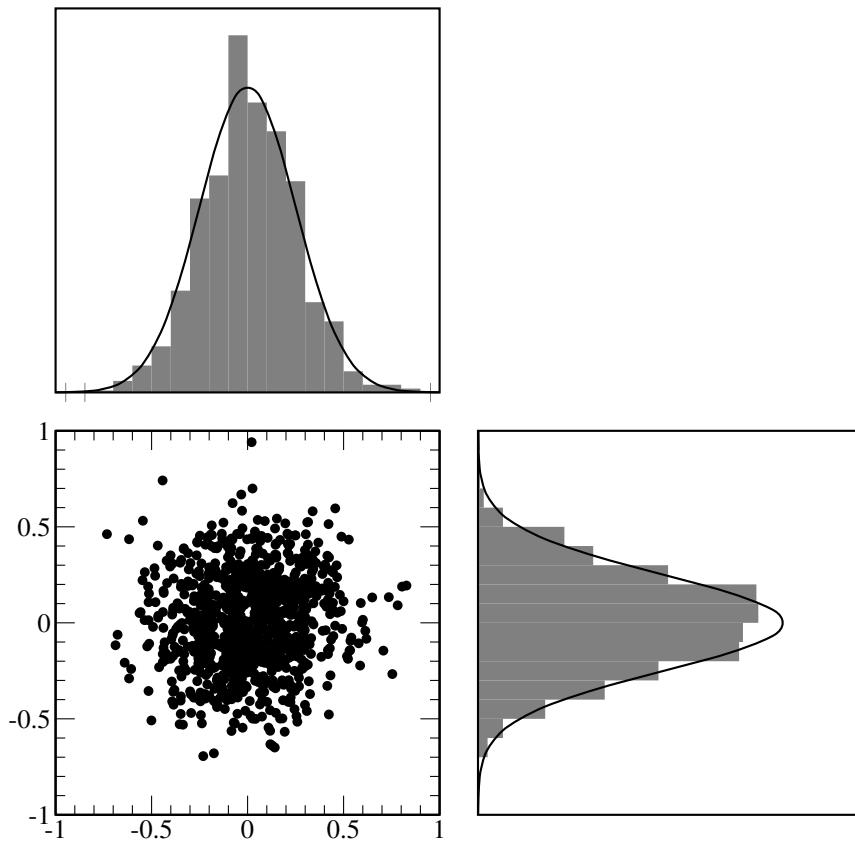
Figure 90: Testing random number generation [random.m4].
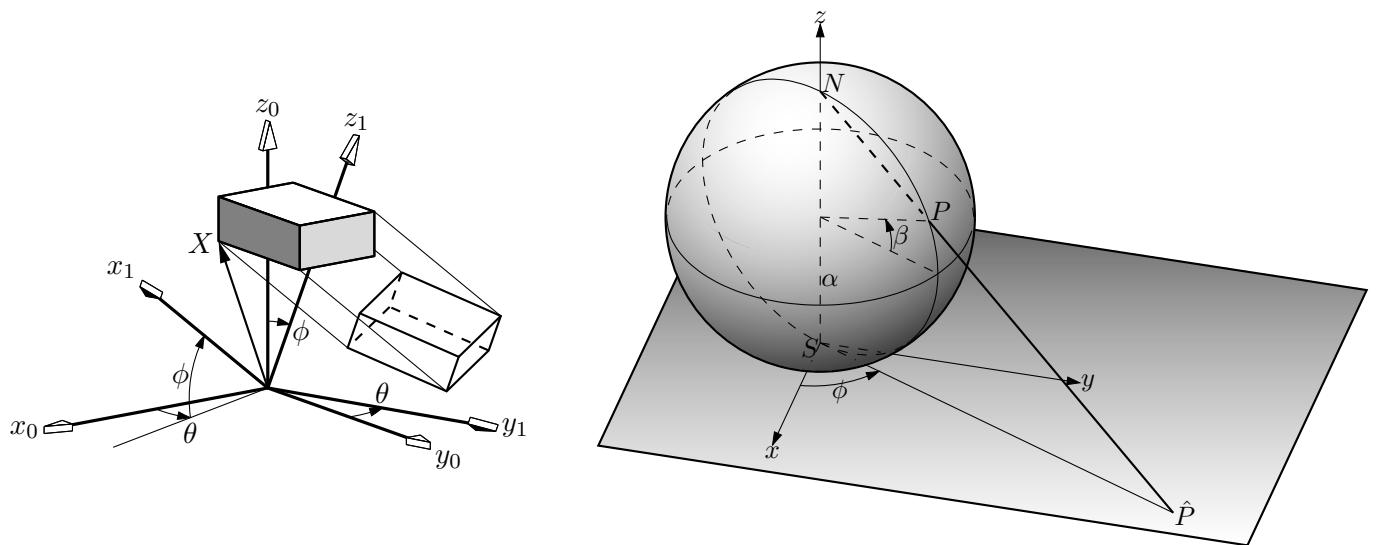
Figure 91: Test of `project` and other `lib3D` macros, showing the projection of a solid onto the y_1, z_1 plane by sighting along the x_1 axis. [`exp.m4`].
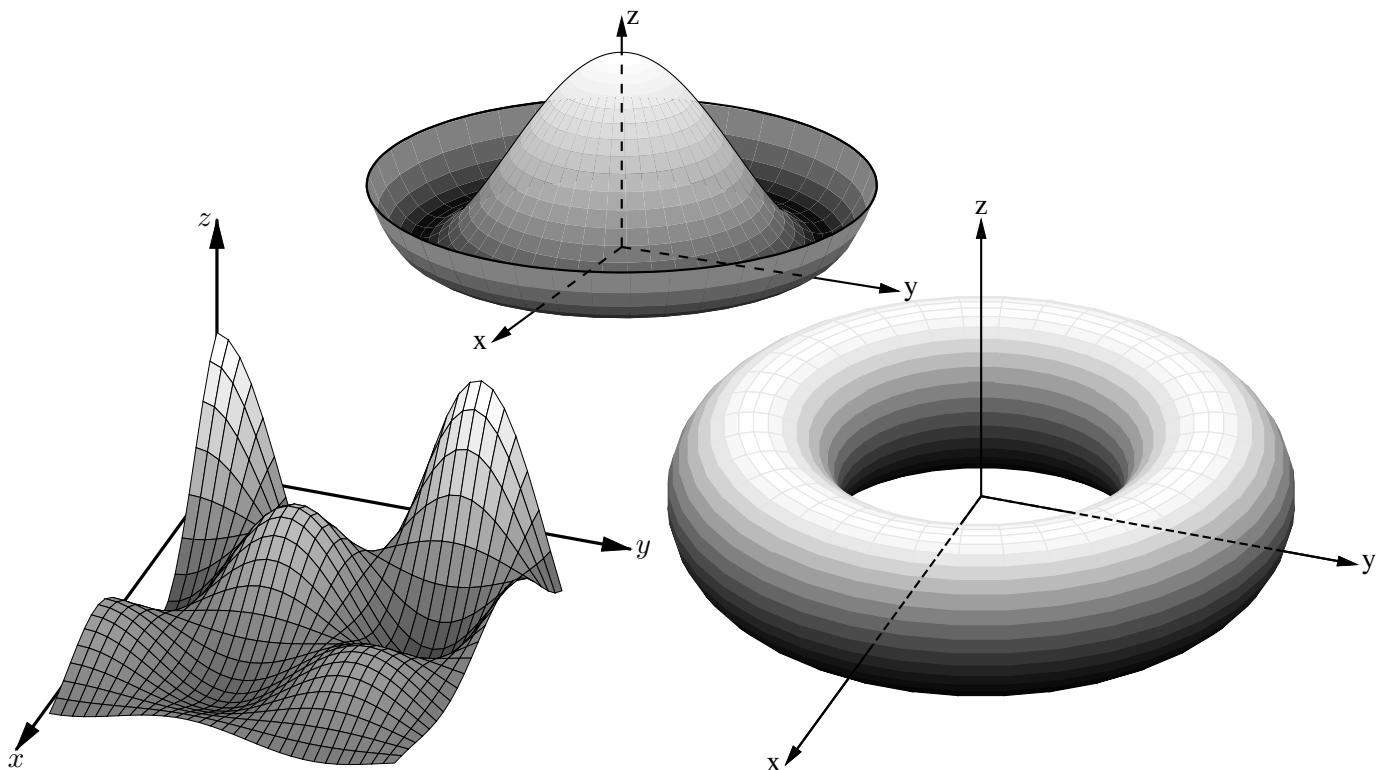

Figure 92: Plotting surfaces using gray scales. A sort algorithm determines plotting order [`graysurf.m4`].
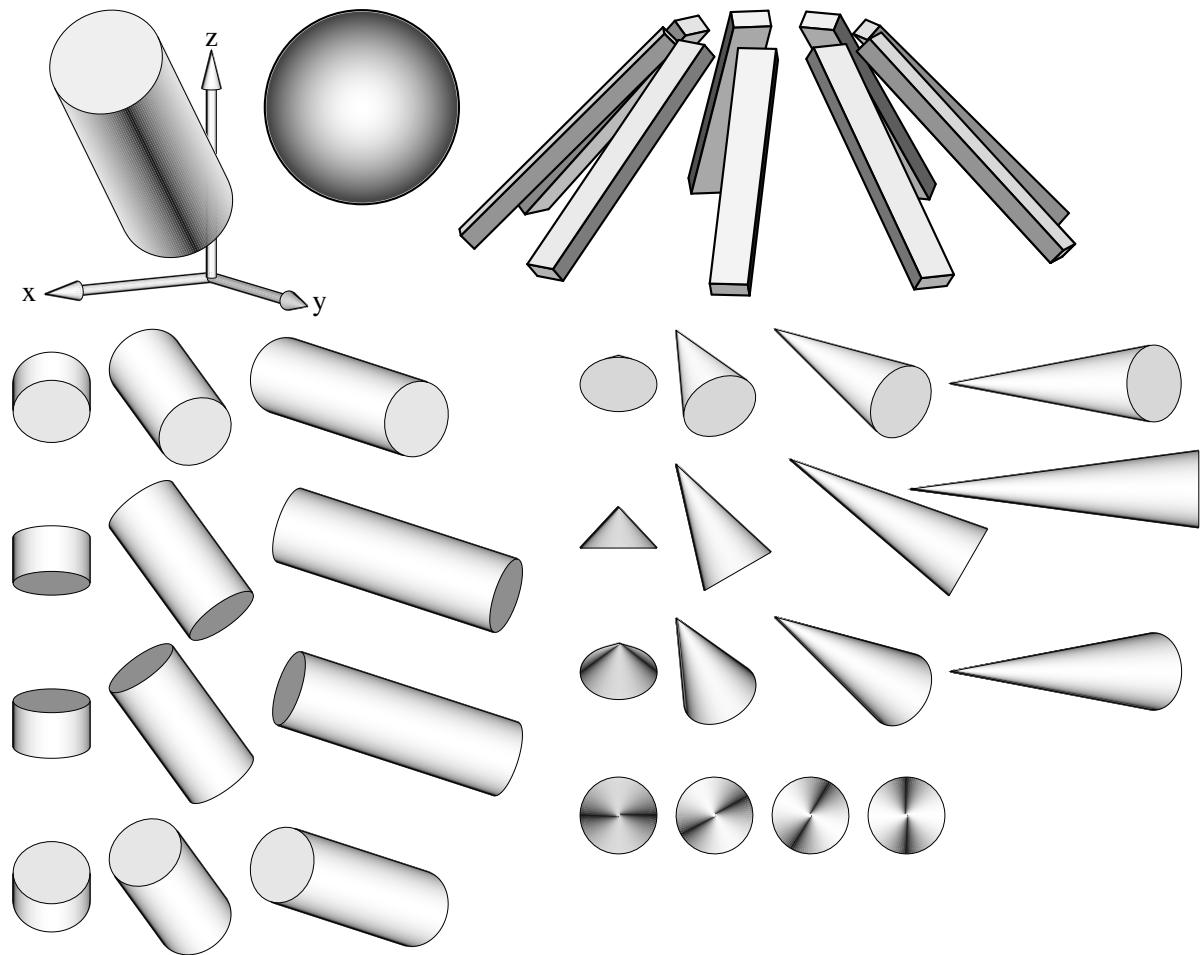
Figure 93: Basic shapes [shapes.m4].
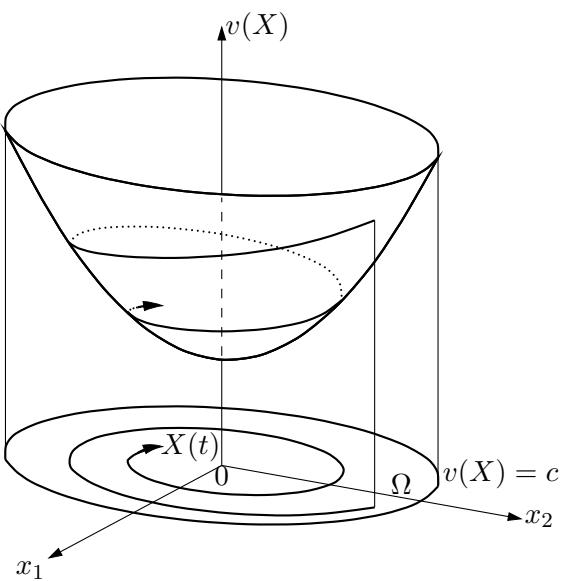
Figure 94: Illustrating a Lyapunov function [Lyap.m4].
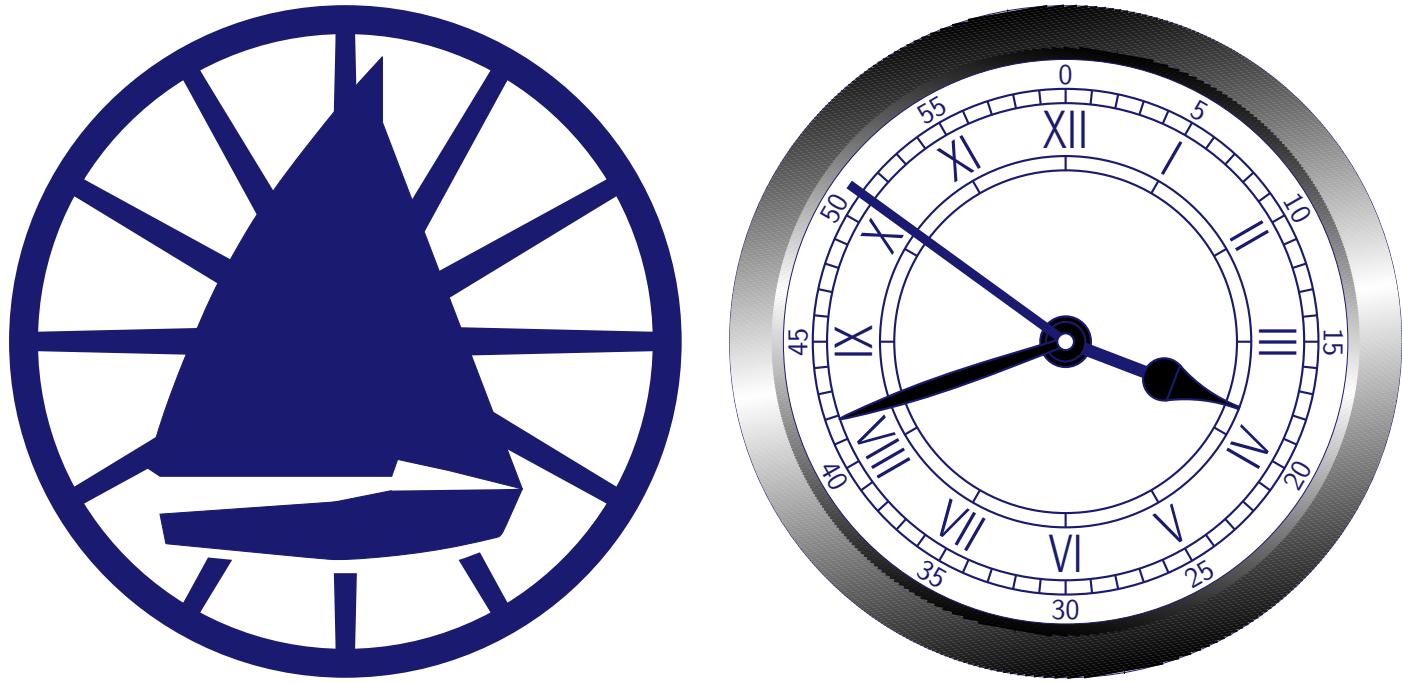
Figure 95: Conestoga Sailing Club (illustrating the filling of arbitrary shapes) and an antique clock face with shading and rotated text [csc.m4].
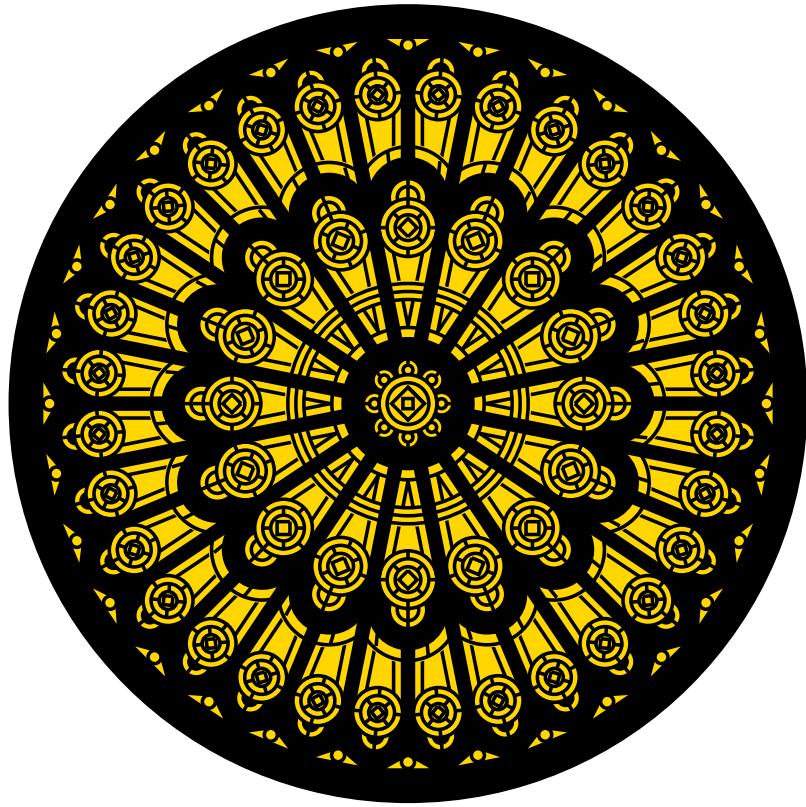
Figure 96: Redrawn from a detail of the set design for the musical *Dracula,* used for testing dpic. This diagram consumes much LATEX main memory but can be produced directly as pdf using dpic -d, as svg using dpic -g, or as postscript using dpic -r since no text formatting is required [rose.m4].
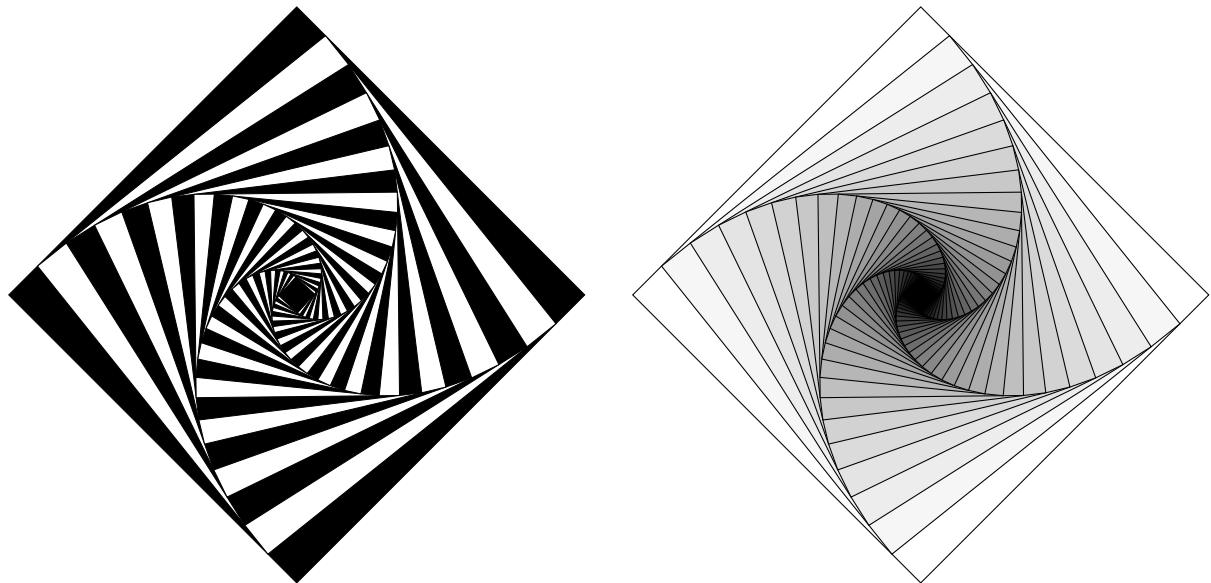
Figure 97: Variations on M. Goossens, S. Rahtz, and F. Mittelbach, *The LaTeX Graphics Companion,* Addison-Wesley 1997, pp. 57-58 [`diamond.m4`].
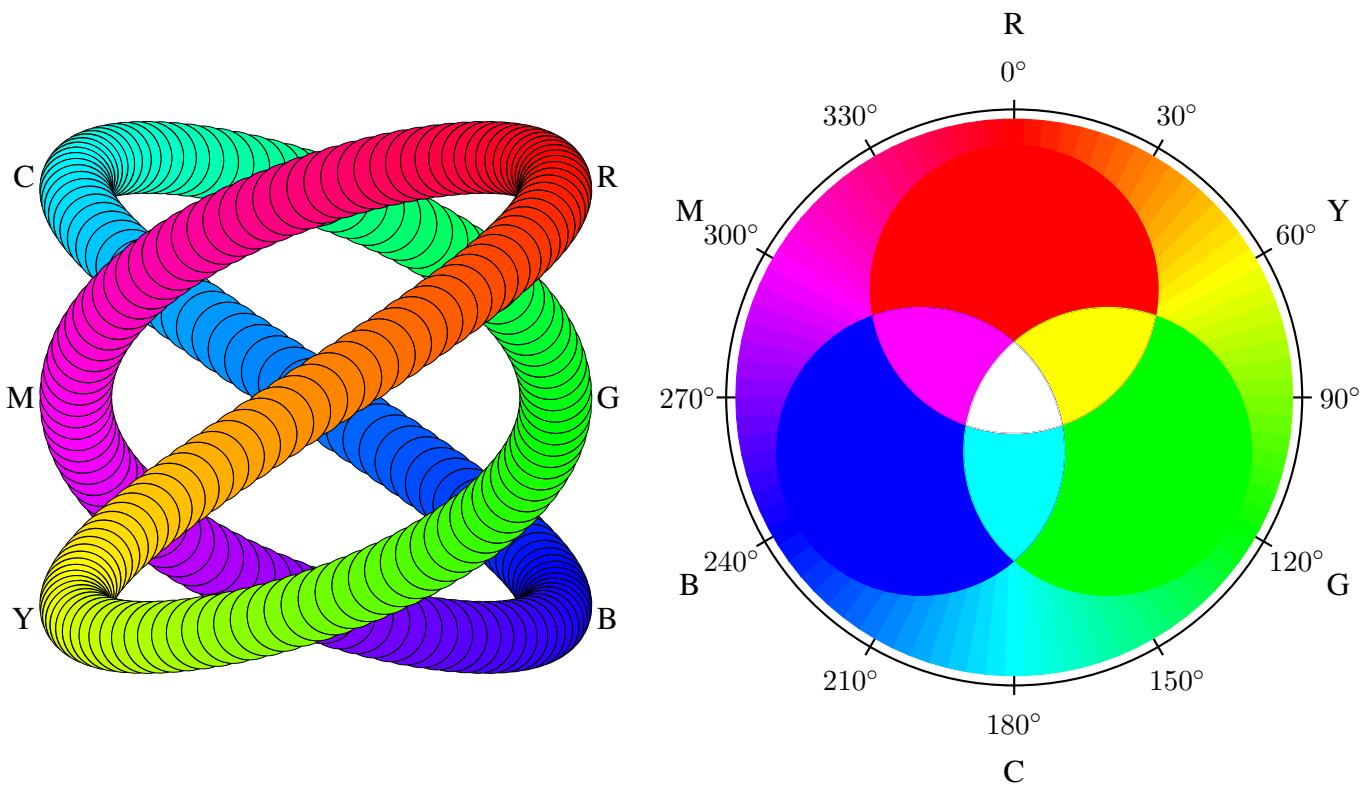
Figure 98: An exercise in calculating RGB colours [`worm.m4`].
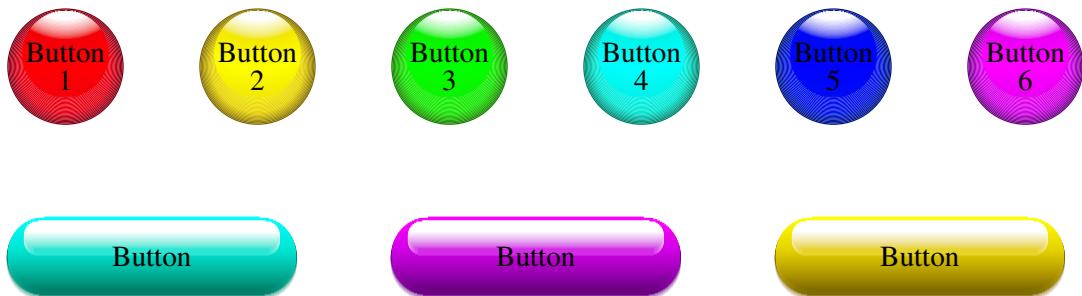
Figure 99: Shading in color [`Buttons.m4`].
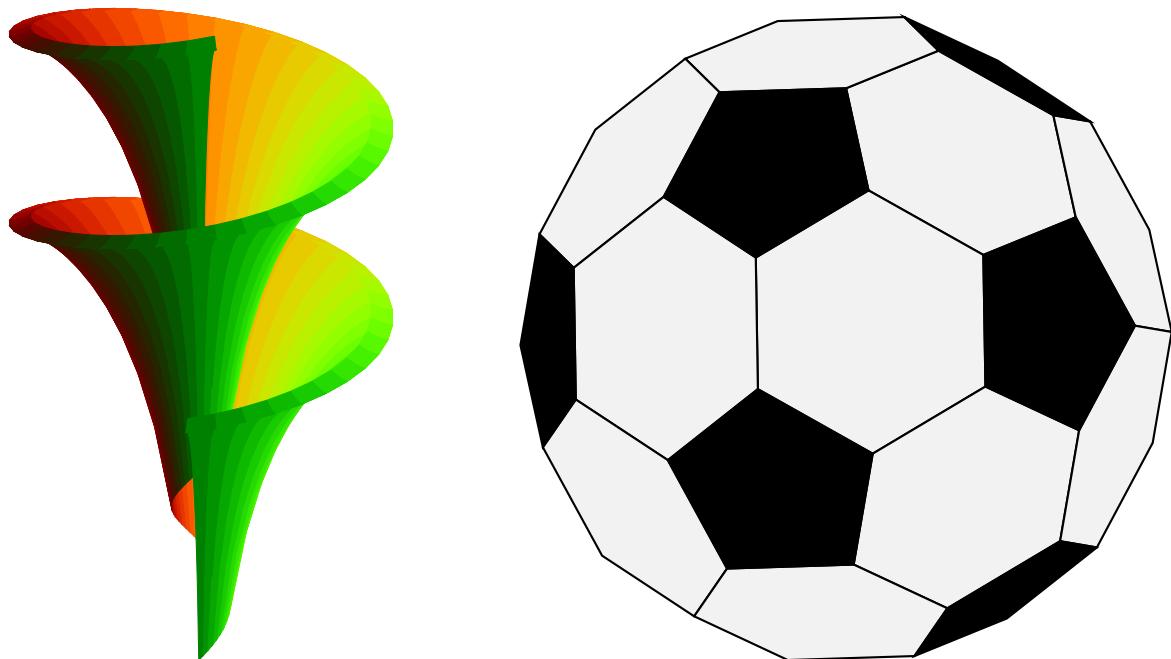
Figure 100: Dini surface and an icosahedron [`Dini.m4`].
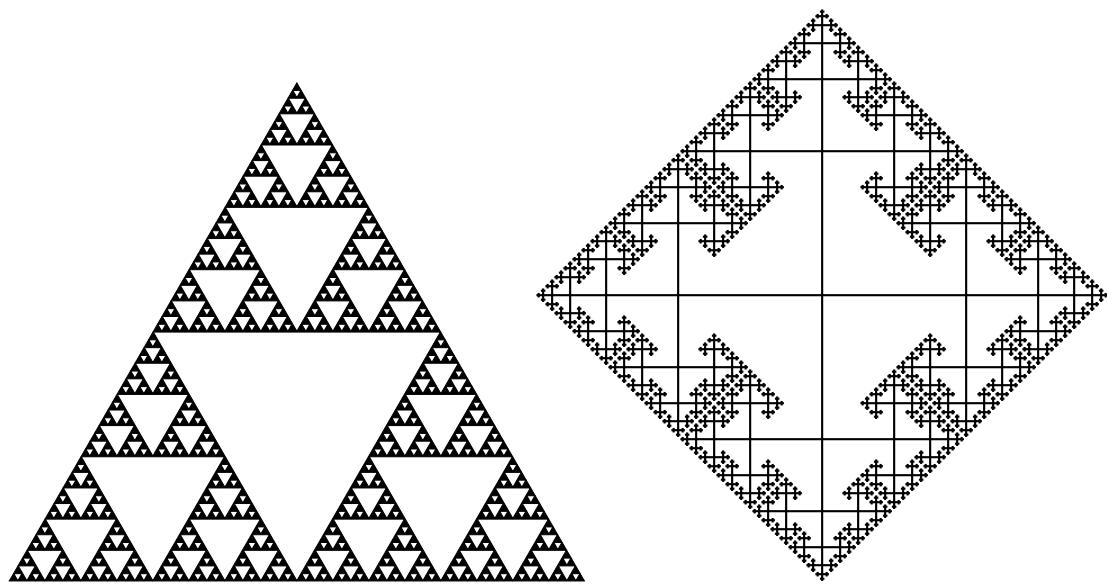
Figure 101: The Sierpinski triangle and a Cayley graph: tests of pic macro recursion [Sierpinski.m4].
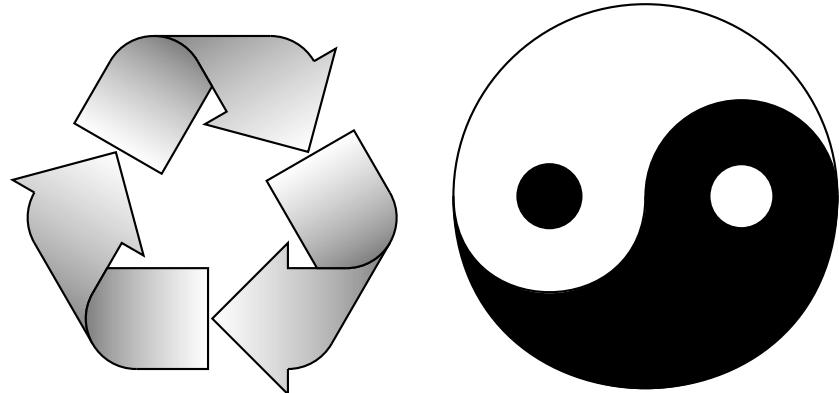

Figure 102: Modest repetition and partial fill [recycle.m4].
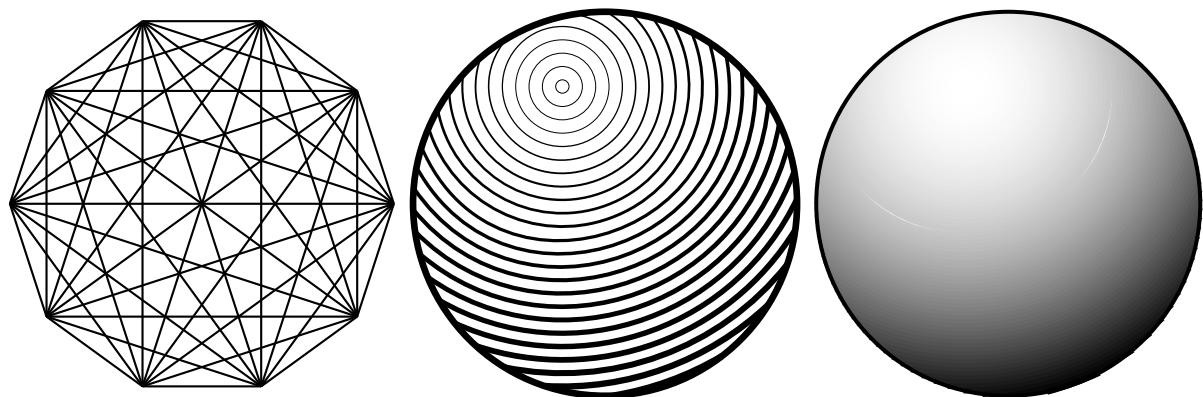
Figure 103: Simple diagrams that are easily drawn by looping [ex15.m4].
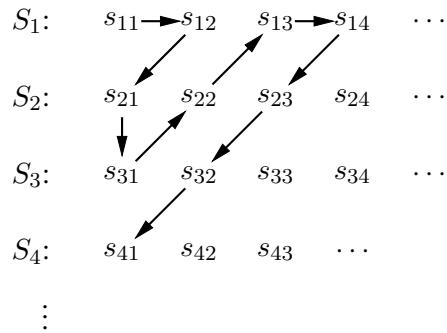
Figure 104: An example of enumeration [Counting.m4].
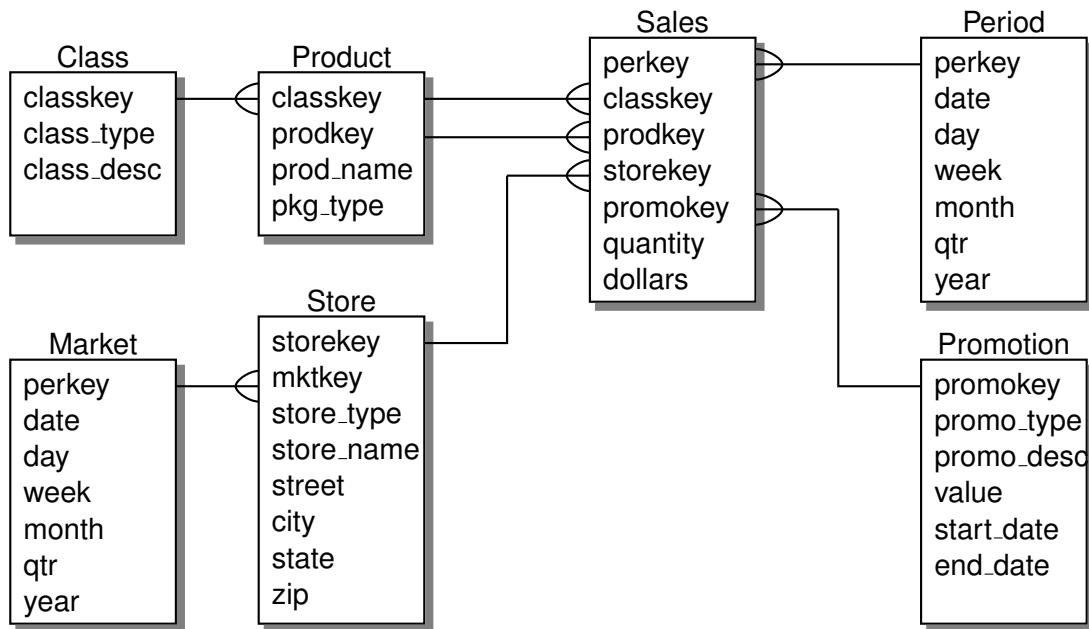
Figure 105: Illustrating shadebox and a custom crowfoot line termination [Crow.m4].
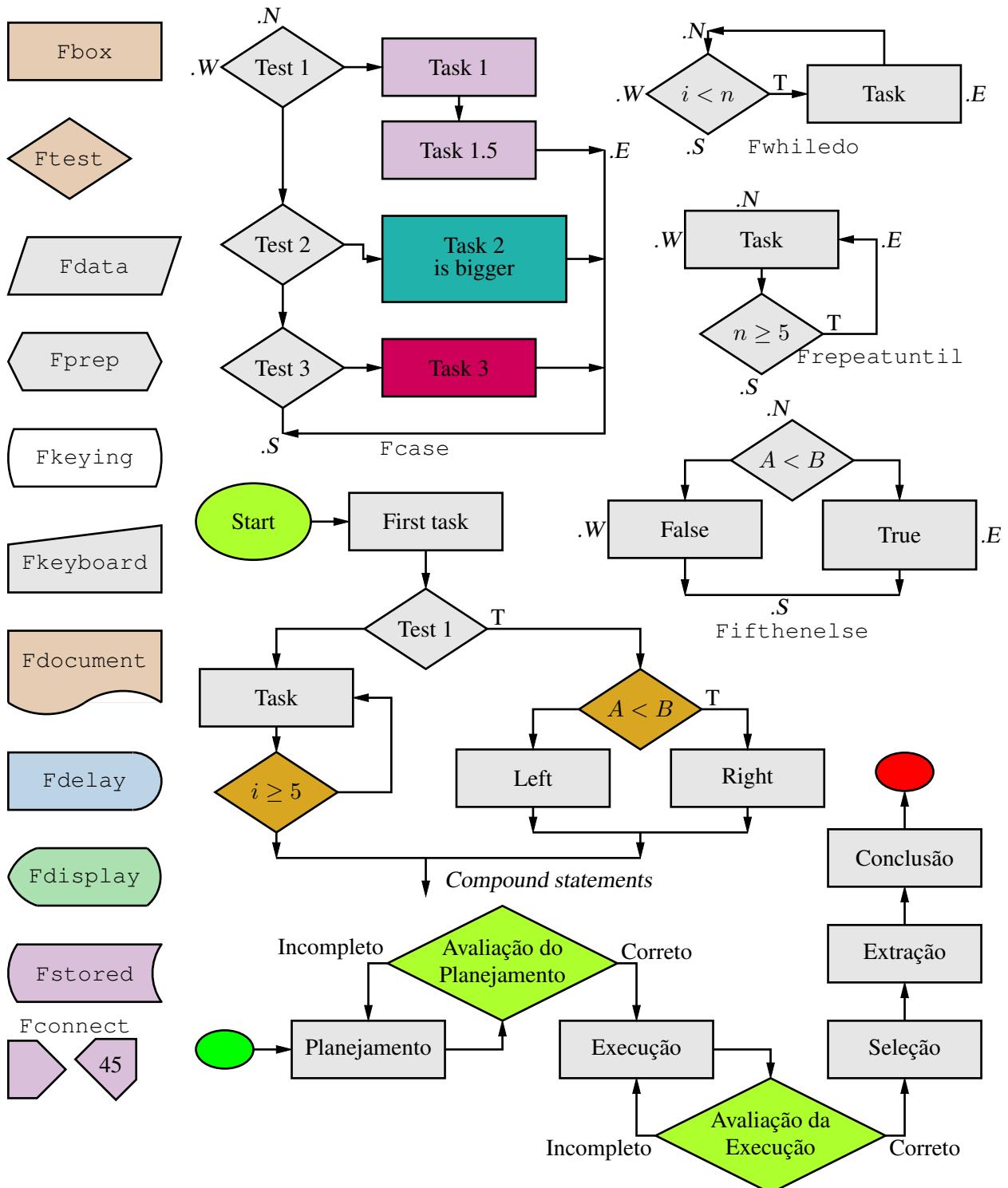
Figure 106: A flowchart sampler [Flow.m4].
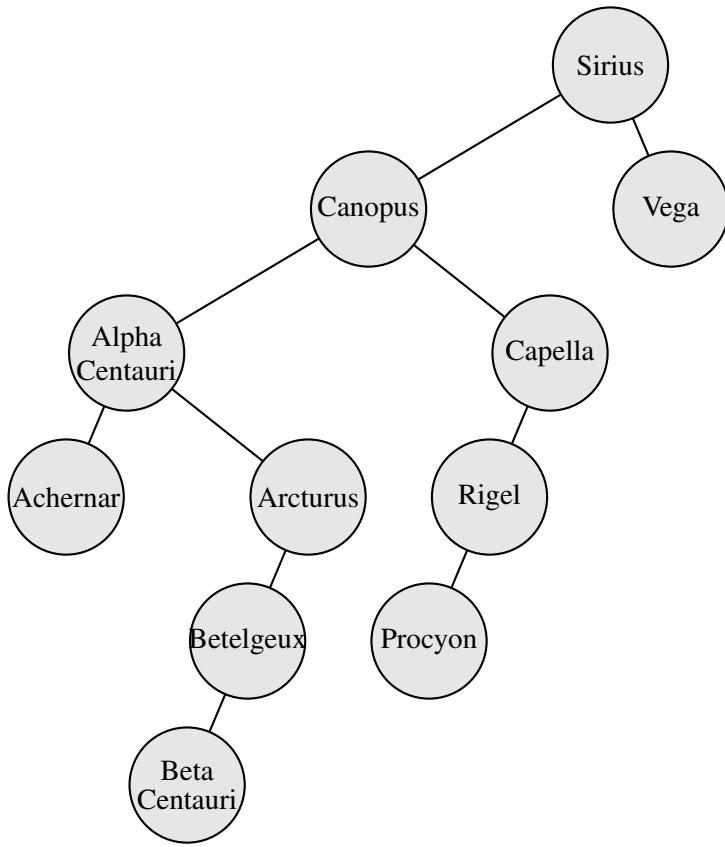
Figure 107: A binary tree [Btree.m4].
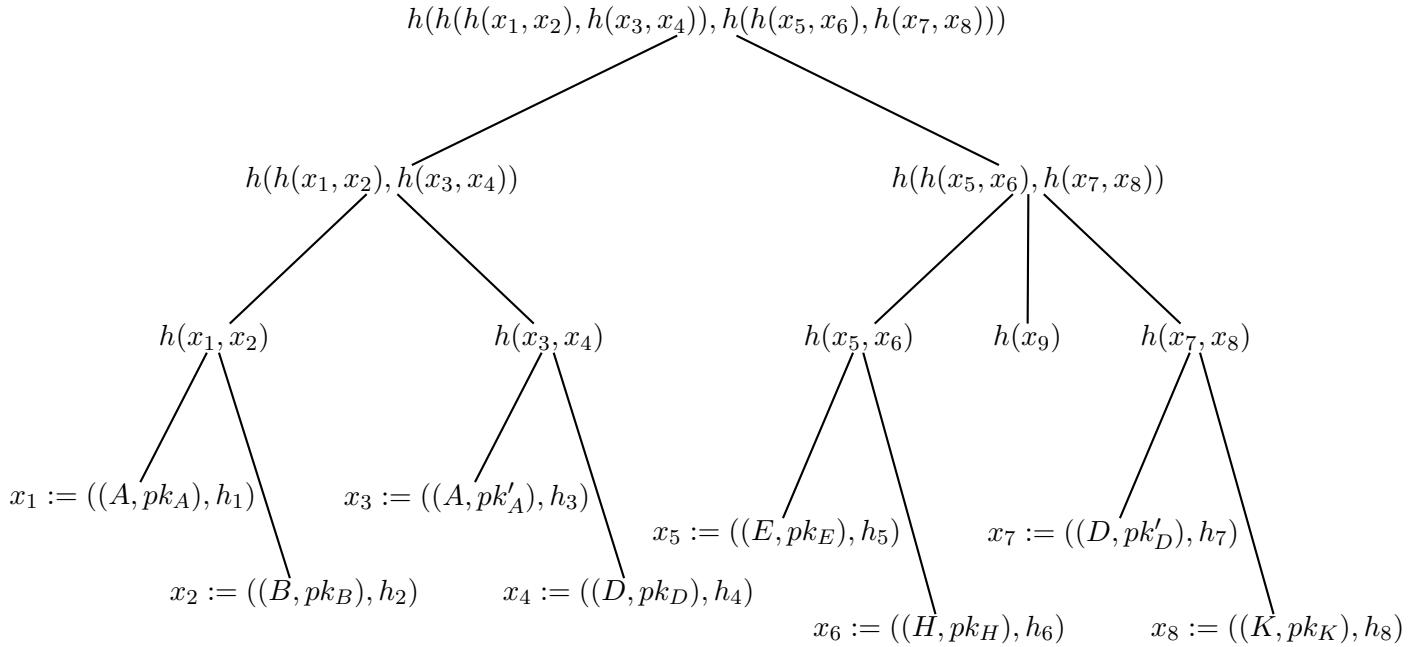
Figure 108: Another tree [Ttree.m4].

Figure 109: Overlaying a figure with line graphics [Incleps.m4].