```
/***************************************************************************\
*                                                                           *
* 6. MECHANICAL/HARDWARE                                                    *
*                                                                           *
\***************************************************************************/

/** \def KINEMATICS_STRAIGHT KINEMATICS_COREXY

  This defines the type of kinematics your printer uses. That's essential!

  Valid values (see dda_kinematics.h):

  KINEMATICS_STRAIGHT
    Motors move axis directions directly. This is the
    traditional type, found in many printers, including
    Mendel, Prusa i3, Mendel90, Ormerod, Mantis.

  KINEMATICS_COREXY
    A bot using CoreXY kinematics. Typical for CoreXY
    are long and crossing toothed belts and a print head
    moving on the X-Y-plane.
*/
#define KINEMATICS_STRAIGHT
//#define KINEMATICS_COREXY

/** \def STEPS_PER_M_X STEPS_PER_M_Y STEPS_PER_M_Z STEPS_PER_M_E
  Steps per meter ( = steps per mm * 1000 ), calculate these values
  appropriate for your machine.

  All numbers are integers, so no decimal point, please :-)

    Valid range: 20 to 4'0960'000 (0.02 to 40960 steps/mm)
*/
#define STEPS_PER_M_X           87489
#define STEPS_PER_M_Y           87489
#define STEPS_PER_M_Z           1280000
#define STEPS_PER_M_E           1600000

/** \def MAXIMUM_FEEDRATE_X MAXIMUM_FEEDRATE_Y MAXIMUM_FEEDRATE_Z
MAXIMUM_FEEDRATE_E
  Used for G0 rapid moves and as a cap for all other feedrates.
*/
#define MAXIMUM_FEEDRATE_X      6000
#define MAXIMUM_FEEDRATE_Y      6000
#define MAXIMUM_FEEDRATE_Z      200
#define MAXIMUM_FEEDRATE_E      2000

/** \def SEARCH_FEEDRATE_X SEARCH_FEEDRATE_Y SEARCH_FEEDRATE_Z
  Used when doing precision endstop search and as default feedrate. No
  SEARCH_FEEDRATE_E, as E can't be searched.
*/
#define SEARCH_FEEDRATE_X       200
#define SEARCH_FEEDRATE_Y       200
#define SEARCH_FEEDRATE_Z       50

/** \def ENDSTOP_CLEARANCE_X ENDSTOP_CLEARANCE_Y ENDSTOP_CLEARANCE_Z

  When hitting an endstop, Teacup properly decelerates instead of doing an
  aprupt stop to save your mechanics. Ineviteably, this means it overshoots
  the endstop trigger point by some distance.

  To deal with this, Teacup adapts homing movement speeds to what your
  endstops can deal with. The higher the allowed acceleration ( = deceleration,
  see #define ACCELERATION) and the more clearance the endstop comes with,
  the faster Teacup will do homing movements.

  Set here how many micrometers (mm * 1000) your endstop allows the carriage
  to overshoot the trigger point. Typically 1000 or 2000 for mechanical
  endstops, more for optical ones. You can set it to zero, in which case
  SEARCH_FEEDRATE_{XYZ} is used, but expect very slow homing movements.

    Units: micrometers
    Sane values: 0 to 20000   (0 to 20 mm)
    Valid range: 0 to 1000000
*/
#define ENDSTOP_CLEARANCE_X     1000
#define ENDSTOP_CLEARANCE_Y     1000
#define ENDSTOP_CLEARANCE_Z     100

/** \def X_MIN X_MAX Y_MIN Y_MAX Z_MIN Z_MAX
  Soft axis limits. Define them to your machine's size relative to what your
  G-code considers to be the origin (typically the bed's center or the bed's
  front left corner).

  Note that relocating the origin at runtime with G92 will also relocate these
  limits.

  Not defining them at all will disable limits checking and make the binary
  about 250 bytes smaller. Enabling only some of them is perfectly fine.

    Units: millimeters
    Sane values: according to printer build room size
    Valid range: -1000.0 to 1000.0
*/
//#define X_MIN                 0.0
//#define X_MAX                 200.0

#define Y_MIN                   0
#define Y_MAX                   200
```

```
//#define Z_MIN                  0.0
//#define Z_MAX                  140.0

/** \def E_ABSOLUTE
  Some G-code creators produce relative length commands for the extruder,
  others absolute ones. G-code using absolute lengths can be recognized when
  there are G92 E0 commands from time to time. If you have G92 E0 in your
  G-code, define this flag.

  This is the startup default and can be changed with M82/M83 while running.
*/
#define E_ABSOLUTE

/** \def HOMING_OPT
  Options for homing movements a user should be able to choose from in configtool. All
  commented out.
*/
//#define HOMING_OPT none
//#define HOMING_OPT x_negative
//#define HOMING_OPT x_positive
//#define HOMING_OPT y_negative
//#define HOMING_OPT y_positive
//#define HOMING_OPT z_negative
//#define HOMING_OPT z_positive

/** \def DEFINE_HOMING
  Order (and number) of homing movements. Up to 4 homing steps are allowed.
  If you don't need even one axis just DEFINE_HOMING(none).
*/
DEFINE_HOMING(x_negative, y_positive, z_negative)

/** \def ACCELERATION_REPRAP ACCELERATION_RAMPING ACCELERATION_TEMPORAL
  Choose optionally one of ACCELERATION_REPRAP, ACCELERATION_RAMPING or
  ACCELERATION_TEMPORAL. With none of them defined, movements are done
  without acceleration. Recommended is ACCELERATION_RAMPING.
*/
//#define ACCELERATION_REPRAP
#define ACCELERATION_RAMPING
//#define ACCELERATION_TEMPORAL

/** \def ACCELERATION
  How fast to accelerate when using ACCELERATION_RAMPING. Start with 10 for
  milling (high precision) or 1000 for printing.

    Units: mm/s^2
    Useful range: 1 to 10'000
*/
#define ACCELERATION            1000
```

```
/** \def LOOKAHEAD
  Define this to enable look-ahead during *ramping* acceleration to smoothly
  transition between moves instead of performing a dead stop every move.
  Enabling look-ahead requires about 3600 bytes of flash memory.
*/
#define LOOKAHEAD

/** \def MAX_JERK_X MAX_JERK_Y MAX_JERK_Z MAX_JERK_E
  When performing look-ahead, we need to decide what an acceptable jerk to the
  mechanics is. Look-ahead attempts to instantly change direction at movement
  crossings, which means instant changes in the speed of the axes participating
  in the movement. Define here how big the speed bumps on each of the axes is
  allowed to be.

  If you want a full stop before and after moving a specific axis, define
  MAX_JERK of this axis to 0. This is often wanted for the Z axis. If you want
  to ignore jerk on an axis, define it to twice the maximum feedrate of this
  axis.

  Having these values too low results in more than neccessary slowdown at
  movement crossings, but is otherwise harmless. Too high values can result
  in stepper motors suddenly stalling. If angles between movements in your
  G-code are small and your printer runs through entire curves full speed,
  there's no point in raising the values.

    Units: mm/min
    Sane values: 0 to 400
    Valid range: 0 to 65535
*/
#define MAX_JERK_X            200
#define MAX_JERK_Y            200
#define MAX_JERK_Z            20
#define MAX_JERK_E            200

/** \def BED_LEVELING
  Define this to enable dynamic bed leveling using the G29 command and
  3-point planar bed mapping. Allows the printer to compensate dynamically
  for a print bed which is flat but is not quite level.
  Enabling bed-leveling requires about 2400 bytes of flash memory.
*/
//#define BED_LEVELING


/****************************************************************************\
*                                                                            *
* 7. MISCELLANEOUS OPTIONS                                                   *
*                                                                            *
\****************************************************************************/

/** \def USE_INTERNAL_PULLUPS
```

```
  Most controller chips feature internal pullup resistors on their input pins,
  which get used for endstops by turning on this switch. Don't turn it on when
  using endstops which need no pull resistor, e.g. optical endstops, because
  pull resistors are counterproductive there.

  One can't use USE_INTERNAL_PULLUPS and USE_INTERNAL_PULLDOWNS at the same
  time, of course.
*/
//#define USE_INTERNAL_PULLUPS

/** \def USE_INTERNAL_PULLDOWNS

  Some controller chips feature internal pulldown resistors on their input
  pins, which get used for endstops by turning on this switch. Don't turn it
  on when using endstops which need no pull resistor, e.g. optical endstops,
  because pull resistors are counterproductive there.

  One can't use USE_INTERNAL_PULLDOWNS and USE_INTERNAL_PULLUPS at the same
  time, of course.
*/
//#define USE_INTERNAL_PULLDOWNS

/** \def Z_AUTODISABLE
  Automatically disable Z axis when not in use. This is useful for printers
  with a self-locking Z axis, e.g. the various Mendel derivates.

  Other printers have a heavy Z axis or a not self-locking spindle. In that
  case you should not activate this.

  This option has no effect on controllers with a common stepper enable pin.
*/
#define Z_AUTODISABLE

/** \def TEMP_HYSTERESIS
  Actual temperature must be target +/- this hysteresis before target
  temperature is considered to be achieved. Also, BANG_BANG tries to stay
  within half of this hysteresis.

    Unit: degree Celsius
*/
#define TEMP_HYSTERESIS         10

/** \def TEMP_RESIDENCY_TIME
  Actual temperature must be close to target (within set temperature
  +- TEMP_HYSTERESIS) for this long before target is achieved (and a M116
  succeeds).

    Unit: seconds
*/
```

```
#define TEMP_RESIDENCY_TIME      60

/** \def TEMP_EWMA

  Smooth noisy temperature sensors. Good hardware shouldn't be noisy. Set to
  1000 for unfiltered data (and a 140 bytes smaller binary).

  Instrument Engineer's Handbook, 4th ed, Vol 2 p126 says values of
  50 to 100 are typical. Smaller is smoother but slower adjusting, larger is
  quicker but rougher. If you need to use this, set the PID parameter to zero
  (M132 S0) to make the PID loop insensitive to noise.

    Valid range: 1 to 1000
*/
#define TEMP_EWMA               1000

/** \def REPORT_TARGET_TEMPS
  With this enabled, M105 commands will return the current temperatures along
  with the target temps, separated by a slash: ok T:xxx.x/xxx.x B:xxx.x/xxx.x
  With this disabled, only temps will be returned: ok T:xxx.x B:xxx.x
  Enabling adds 78 bytes to the image.
*/
#define REPORT_TARGET_TEMPS

/** \def HEATER_SANITY_CHECK
  Check if heater responds to changes in target temperature, disable and spit
  errors if not largely untested, please comment in forum if this works, or
  doesn't work for you!
*/
//#define HEATER_SANITY_CHECK

/** \def EECONFIG
  Enable EEPROM configuration storage.

  Enabled by default. Commenting this out makes the binary several hundred
  bytes smaller, so you might want to disable EEPROM storage on small MCUs,
  like the ATmega168.
*/
#define EECONFIG

/** \def BANG_BANG
  Drops PID loop from heater control, reduces code size significantly
  (1300 bytes!).
*/
//#define BANG_BANG

/** \def BANG_BANG_ON
  PWM value for Bang Bang 'on'.
*/
//#define BANG_BANG_ON            200
```

```
/** \def BANG_BANG_OFF
  PWM value for Bang Bang 'off'.
*/
//#define BANG_BANG_OFF           45

/** \def MOVEBUFFER_SIZE
  Move buffer size, in number of moves.

  Note that each move takes a fair chunk of ram (107 bytes as of this writing),
  so don't make the buffer too big. However, a larger movebuffer will probably
  help with lots of short consecutive moves, as each move takes a bunch of
  math (hence time) to set up so a longer buffer allows more of the math to
  be done during preceding longer moves.
*/
#define MOVEBUFFER_SIZE          16

/** \def DC_EXTRUDER DC_EXTRUDER_PWM
  If you have a DC motor extruder, configure it as a "heater" above and define
  this value as the index or name. You probably also want to comment out
  E_STEP_PIN and E_DIR_PIN in the Pinouts section above.
*/
//#define DC_EXTRUDER             HEATER_motor
//#define DC_EXTRUDER_PWM         180

/** \def USE_WATCHDOG
  Teacup implements a watchdog, which has to be reset every 250ms or it will
  reboot the controller. As rebooting (and letting the GCode sending
  application trying to continue the build with a then different Home point)
  is probably even worse than just hanging, and there is no better restore
  code in place, this is disabled for now.
*/
//#define USE_WATCHDOG

/** \def TH_COUNT
  Temperature history count. This is how many temperature readings to keep in
  order to calculate derivative in PID loop higher values make PID derivative
  term more stable at the expense of reaction time.
*/
#define TH_COUNT                 8

/** \def FAST_PWM
  Teacup offers two PWM frequencies, 76(61) Hz and 78000(62500) Hz on a
  20(16) MHz electronics. The slower one is the default, as it's the safer
  choice and reduces MOSFET heating. Drawback is, in a quiet environment you
  might notice the heaters and your power supply humming.

  Uncomment this option if you want to get rid of this humming and can afford
  a hotter MOSFET or want faster PWM for other reasons.
  See also: http://reprap.org/wiki/Gen7_Research#MOSFET_heat_and_PWM
*/
//#define FAST_PWM

/** \def PID_SCALE
  This is the scaling of internally stored PID values. 1024L is a good value.
*/
#define PID_SCALE                 1024L

/** \def ENDSTOP_STEPS
  Number of steps to run into the endstops intentionally. As endstops trigger
  false alarm sometimes, Teacup debounces them by counting a number of
  consecutive positives.

  Use 4 or less for reliable endstops, 8 or even more for flaky ones.

    Valid range: 1...255.
*/
#define ENDSTOP_STEPS             4

/** \def CANNED_CYCLE
  G-code commands in this string will be executed over and over again, without
  user interaction or even a serial connection. It's purpose is e.g. for
  exhibitions or when using Teacup for other purposes than printing. You can
  add any G-code supported by Teacup.

  Note: don't miss these newlines (\n) and backslashes (\).
*/
/*
#define CANNED_CYCLE "G1 X100 F3000\n" \
"G4 P500\n" \
"G1 X0\n" \
"G4 P500\n"
*/
```