```
/**
 * Marlin 3D Printer Firmware
 * Copyright (C) 2016 MarlinFirmware [https://github.com/MarlinFirmware/Marlin]
 *
 * Based on Sprinter and grbl.
 * Copyright (C) 2011 Camiel Gubbels / Erik van der Zalm
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

/**
 * Configuration_adv.h
 *
 * Advanced settings.
 * Only change these if you know exactly what you're doing.
 * Some of these settings can damage your printer if improperly set!
 *
 * Basic settings can be found in Configuration.h
 *
 */
#ifndef CONFIGURATION_ADV_H
#define CONFIGURATION_ADV_H
#define CONFIGURATION_ADV_H_VERSION 010100

// @section temperature

//===========================================================================
//=============================Thermal Settings  ============================
//===========================================================================

#if DISABLED(PIDTEMPBED)
  #define BED_CHECK_INTERVAL 5000 // ms between checks in bang-bang control
  #if ENABLED(BED_LIMIT_SWITCHING)
    #define BED_HYSTERESIS 2 // Only disable heating if T>target+BED_HYSTERESIS and enable heating if
T>target-BED_HYSTERESIS
  #endif
#endif

/**
 * Thermal Protection protects your printer from damage and fire if a
 * thermistor falls out or temperature sensors fail in any way.
 *
 * The issue: If a thermistor falls out or a temperature sensor fails,
 * Marlin can no longer sense the actual temperature. Since a disconnected
 * thermistor reads as a low temperature, the firmware will keep the heater on.
 *
 * The solution: Once the temperature reaches the target, start observing.
 * If the temperature stays too far below the target (hysteresis) for too long (period),
 * the firmware will halt the machine as a safety precaution.
 *
 * If you get false positives for "Thermal Runaway" increase THERMAL_PROTECTION_HYSTERESIS and/or
THERMAL_PROTECTION_PERIOD
 */
#if ENABLED(THERMAL_PROTECTION_HOTENDS)
  #define THERMAL_PROTECTION_PERIOD 60        // Seconds
  #define THERMAL_PROTECTION_HYSTERESIS 10    // Degrees Celsius

  /**
   * Whenever an M104 or M109 increases the target temperature the firmware will wait for the
   * WATCH_TEMP_PERIOD to expire, and if the temperature hasn't increased by WATCH_TEMP_INCREASE
   * degrees, the machine is halted, requiring a hard reset. This test restarts with any M104/M109,
   * but only if the current temperature is far enough below the target for a reliable test.
   *
   * If you get false positives for "Heating failed" increase WATCH_TEMP_PERIOD and/or decrease
WATCH_TEMP_INCREASE
   * WATCH_TEMP_INCREASE should not be below 2.
   */
  #define WATCH_TEMP_PERIOD 20                // Seconds
  #define WATCH_TEMP_INCREASE 2               // Degrees Celsius
```

```
#endif

/**
 * Thermal Protection parameters for the bed are just as above for hotends.
 */
#if ENABLED(THERMAL_PROTECTION_BED)
  #define THERMAL_PROTECTION_BED_PERIOD 20    // Seconds
  #define THERMAL_PROTECTION_BED_HYSTERESIS 2 // Degrees Celsius

  /**
   * Whenever an M140 or M190 increases the target temperature the firmware will wait for the
   * WATCH_BED_TEMP_PERIOD to expire, and if the temperature hasn't increased by WATCH_BED_TEMP_INCREASE
   * degrees, the machine is halted, requiring a hard reset. This test restarts with any M140/M190,
   * but only if the current temperature is far enough below the target for a reliable test.
   *
   * If you get too many "Heating failed" errors, increase WATCH_BED_TEMP_PERIOD and/or decrease
   * WATCH_BED_TEMP_INCREASE. (WATCH_BED_TEMP_INCREASE should not be below 2.)
   */
  #define WATCH_BED_TEMP_PERIOD 60                // Seconds
  #define WATCH_BED_TEMP_INCREASE 2               // Degrees Celsius
#endif

#if ENABLED(PIDTEMP)
  // this adds an experimental additional term to the heating power, proportional to the extrusion speed.
  // if Kc is chosen well, the additional required power due to increased melting should be compensated.
  //#define PID_EXTRUSION_SCALING
  #if ENABLED(PID_EXTRUSION_SCALING)
    #define DEFAULT_Kc (100) //heating power=Kc*(e_speed)
    #define LPQ_MAX_LEN 50
  #endif
#endif

/**
 * Automatic Temperature:
 * The hotend target temperature is calculated by all the buffered lines of gcode.
 * The maximum buffered steps/sec of the extruder motor is called "se".
 * Start autotemp mode with M109 S<mintemp> B<maxtemp> F<factor>
 * The target temperature is set to mintemp+factor*se[steps/sec] and is limited by
 * mintemp and maxtemp. Turn this off by executing M109 without F*
 * Also, if the temperature is set to a value below mintemp, it will not be changed by autotemp.
 * On an Ultimaker, some initial testing worked with M109 S215 B260 F1 in the start.gcode
 */
#define AUTOTEMP
#if ENABLED(AUTOTEMP)
  #define AUTOTEMP_OLDWEIGHT 0.98
#endif

// Show Temperature ADC value
// Enable for M105 to include ADC values read from temperature sensors.
//#define SHOW_TEMP_ADC_VALUES

/**
 * High Temperature Thermistor Support
 *
 * Thermistors able to support high temperature tend to have a hard time getting
 * good readings at room and lower temperatures. This means HEATER_X_RAW_LO_TEMP
 * will probably be caught when the heating element first turns on during the
 * preheating process, which will trigger a min_temp_error as a safety measure
 * and force stop everything.
 * To circumvent this limitation, we allow for a preheat time (during which,
 * min_temp_error won't be triggered) and add a min_temp buffer to handle
 * aberrant readings.
 *
 * If you want to enable this feature for your hotend thermistor(s)
 * uncomment and set values > 0 in the constants below
 */

// The number of consecutive low temperature errors that can occur
// before a min_temp_error is triggered. (Shouldn't be more than 10.)
//#define MAX_CONSECUTIVE_LOW_TEMPERATURE_ERROR_ALLOWED 0

// The number of milliseconds a hotend will preheat before starting to check
// the temperature. This value should NOT be set to the time it takes the
// hot end to reach the target temperature, but the time it takes to reach
// the minimum temperature your thermistor can read. The lower the better/safer.
// This shouldn't need to be more than 30 seconds (30000)
//#define MILLISECONDS_PREHEAT_TIME 0

// @section extruder

// Extruder runout prevention.
```

```
// If the machine is idle and the temperature over MINTEMP
// then extrude some filament every couple of SECONDS.
//#define EXTRUDER_RUNOUT_PREVENT
#if ENABLED(EXTRUDER_RUNOUT_PREVENT)
  #define EXTRUDER_RUNOUT_MINTEMP 190
  #define EXTRUDER_RUNOUT_SECONDS 30
  #define EXTRUDER_RUNOUT_SPEED 1500  // mm/m
  #define EXTRUDER_RUNOUT_EXTRUDE 5   // mm
#endif

// @section temperature

//These defines help to calibrate the AD595 sensor in case you get wrong temperature measurements.
//The measured temperature is defined as "actualTemp = (measuredTemp * TEMP_SENSOR_AD595_GAIN) +
TEMP_SENSOR_AD595_OFFSET"
#define TEMP_SENSOR_AD595_OFFSET 0.0
#define TEMP_SENSOR_AD595_GAIN   1.0

/**
 * Controller Fan
 * To cool down the stepper drivers and MOSFETs.
 *
 * The fan will turn on automatically whenever any stepper is enabled
 * and turn off after a set period after all steppers are turned off.
 */
#define USE_CONTROLLER_FAN
#if ENABLED(USE_CONTROLLER_FAN)
  #define CONTROLLER_FAN_PIN 11  // Set a custom pin for the controller fan
  #define CONTROLLERFAN_SECS 60          // Duration in seconds for the fan to run after all motors are
disabled
  #define CONTROLLERFAN_SPEED 255        // 255 == full speed
#endif

// When first starting the main fan, run it at full speed for the
// given number of milliseconds.  This gets the fan spinning reliably
// before setting a PWM value. (Does not work with software PWM for fan on Sanguinololu)
//#define FAN_KICKSTART_TIME 100

// This defines the minimal speed for the main fan, run in PWM mode
// to enable uncomment and set minimal PWM speed for reliable running (1-255)
// if fan speed is [1 - (FAN_MIN_PWM-1)] it is set to FAN_MIN_PWM
//#define FAN_MIN_PWM 50

// @section extruder

/**
 * Extruder cooling fans
 *
 * Extruder auto fans automatically turn on when their extruders'
 * temperatures go above EXTRUDER_AUTO_FAN_TEMPERATURE.
 *
 * Your board's pins file specifies the recommended pins. Override those here
 * or set to -1 to disable completely.
 *
 * Multiple extruders can be assigned to the same pin in which case
 * the fan will turn on when any selected extruder is above the threshold.
 */
#define E0_AUTO_FAN_PIN 9
#define E1_AUTO_FAN_PIN -1
#define E2_AUTO_FAN_PIN -1
#define E3_AUTO_FAN_PIN -1
#define E4_AUTO_FAN_PIN -1
#define EXTRUDER_AUTO_FAN_TEMPERATURE 50
#define EXTRUDER_AUTO_FAN_SPEED   255  // == full speed

/**
 * Part-Cooling Fan Multiplexer
 *
 * This feature allows you to digitally multiplex the fan output.
 * The multiplexer is automatically switched at tool-change.
 * Set FANMUX[012]_PINs below for up to 2, 4, or 8 multiplexed fans.
 */
#define FANMUX0_PIN -1
#define FANMUX1_PIN -1
#define FANMUX2_PIN -1

/**
 * M355 Case Light on-off / brightness
 */
//#define CASE_LIGHT_ENABLE
#if ENABLED(CASE_LIGHT_ENABLE)
```

```
  //#define CASE_LIGHT_PIN 4                  // Override the default pin if needed
  #define INVERT_CASE_LIGHT false             // Set true if Case Light is ON when pin is LOW
  #define CASE_LIGHT_DEFAULT_ON true          // Set default power-up state on
  #define CASE_LIGHT_DEFAULT_BRIGHTNESS 105   // Set default power-up brightness (0-255, requires PWM
pin)
  //#define MENU_ITEM_CASE_LIGHT              // Add a Case Light option to the LCD main menu
#endif

//===========================================================================
//============================ Mechanical Settings ==========================
//===========================================================================

// @section homing

// If you want endstops to stay on (by default) even when not homing
// enable this option. Override at any time with M120, M121.
//#define ENDSTOPS_ALWAYS_ON_DEFAULT

// @section extras

//#define Z_LATE_ENABLE // Enable Z the last moment. Needed if your Z driver overheats.

// Dual X Steppers
// Uncomment this option to drive two X axis motors.
// The next unused E driver will be assigned to the second X stepper.
//#define X_DUAL_STEPPER_DRIVERS
#if ENABLED(X_DUAL_STEPPER_DRIVERS)
  // Set true if the two X motors need to rotate in opposite directions
  #define INVERT_X2_VS_X_DIR true
#endif

// Dual Y Steppers
// Uncomment this option to drive two Y axis motors.
// The next unused E driver will be assigned to the second Y stepper.
//#define Y_DUAL_STEPPER_DRIVERS
#if ENABLED(Y_DUAL_STEPPER_DRIVERS)
  // Set true if the two Y motors need to rotate in opposite directions
  #define INVERT_Y2_VS_Y_DIR true
#endif

// A single Z stepper driver is usually used to drive 2 stepper motors.
// Uncomment this option to use a separate stepper driver for each Z axis motor.
// The next unused E driver will be assigned to the second Z stepper.
#define Z_DUAL_STEPPER_DRIVERS

#if ENABLED(Z_DUAL_STEPPER_DRIVERS)

  // Z_DUAL_ENDSTOPS is a feature to enable the use of 2 endstops for both Z steppers - Let's call them Z
stepper and Z2 stepper.
  // That way the machine is capable to align the bed during home, since both Z steppers are homed.
  // There is also an implementation of M666 (software endstops adjustment) to this feature.
  // After Z homing, this adjustment is applied to just one of the steppers in order to align the bed.
  // One just need to home the Z axis and measure the distance difference between both Z axis and apply
the math: Z adjust = Z - Z2.
  // If the Z stepper axis is closer to the bed, the measure Z > Z2 (yes, it is.. think about it) and the
Z adjust would be positive.
  // Play a little bit with small adjustments (0.5mm) and check the behaviour.
  // The M119 (endstops report) will start reporting the Z2 Endstop as well.

  #define Z_DUAL_ENDSTOPS

  #if ENABLED(Z_DUAL_ENDSTOPS)
    #define Z2_USE_ENDSTOP _ZMAX_
    #define Z_DUAL_ENDSTOPS_ADJUSTMENT  0  // Use M666 to determine/test this value
  #endif

#endif // Z_DUAL_STEPPER_DRIVERS

// Enable this for dual x-carriage printers.
// A dual x-carriage design has the advantage that the inactive extruder can be parked which
// prevents hot-end ooze contaminating the print. It also reduces the weight of each x-carriage
// allowing faster printing speeds. Connect your X2 stepper to the first unused E plug.
//#define DUAL_X_CARRIAGE
#if ENABLED(DUAL_X_CARRIAGE)
  // Configuration for second X-carriage
  // Note: the first x-carriage is defined as the x-carriage which homes to the minimum endstop;
  // the second x-carriage always homes to the maximum endstop.
  #define X2_MIN_POS 80     // set minimum to ensure second x-carriage doesn't hit the parked first X-
carriage
  #define X2_MAX_POS 353    // set maximum to the distance between toolheads when both heads are homed
  #define X2_HOME_DIR 1     // the second X-carriage always homes to the maximum endstop position
```

```
  #define X2_HOME_POS X2_MAX_POS // default home position is the maximum carriage position
      // However: In this mode the HOTEND_OFFSET_X value for the second extruder provides a software
      // override for X2_HOME_POS. This also allow recalibration of the distance between the two endstops
      // without modifying the firmware (through the "M218 T1 X???" command).
      // Remember: you should set the second extruder x-offset to 0 in your slicer.

  // There are a few selectable movement modes for dual x-carriages using M605 S<mode>
  //    Mode 0 (DXC_FULL_CONTROL_MODE): Full control. The slicer has full control over both x-carriages
and can achieve optimal travel results
  //                                    as long as it supports dual x-carriages. (M605 S0)
  //    Mode 1 (DXC_AUTO_PARK_MODE)   : Auto-park mode. The firmware will automatically park and unpark
the x-carriages on tool changes so
  //                                    that additional slicer support is not required. (M605 S1)
  //    Mode 2 (DXC_DUPLICATION_MODE) : Duplication mode. The firmware will transparently make the second
x-carriage and extruder copy all
  //                                    actions of the first x-carriage. This allows the printer to print
2 arbitrary items at
  //                                    once. (2nd extruder x offset and temp offset are set using: M605
S2 [Xnnn] [Rmmm])

  // This is the default power-up mode which can be later using M605.
  #define DEFAULT_DUAL_X_CARRIAGE_MODE DXC_FULL_CONTROL_MODE

  // Default settings in "Auto-park Mode"
  #define TOOLCHANGE_PARK_ZLIFT   0.2      // the distance to raise Z axis when parking an extruder
  #define TOOLCHANGE_UNPARK_ZLIFT 1        // the distance to raise Z axis when unparking an extruder

  // Default x offset in duplication mode (typically set to half print bed width)
  #define DEFAULT_DUPLICATION_X_OFFSET 100

#endif // DUAL_X_CARRIAGE

// Activate a solenoid on the active extruder with M380. Disable all with M381.
// Define SOL0_PIN, SOL1_PIN, etc., for each extruder that has a solenoid.
//#define EXT_SOLENOID

// @section homing

//homing hits the endstop, then retracts by this distance, before it tries to slowly bump again:
#define X_HOME_BUMP_MM 5
#define Y_HOME_BUMP_MM 5
#define Z_HOME_BUMP_MM 2
#define HOMING_BUMP_DIVISOR {2, 2, 4}  // Re-Bump Speed Divisor (Divides the Homing Feedrate)
//#define QUICK_HOME  //if this is defined, if both x and y are to be homed, a diagonal move will be
performed initially.

// When G28 is called, this option will make Y home before X
//#define HOME_Y_BEFORE_X

// @section machine

#define AXIS_RELATIVE_MODES {false, false, false, false}

// Allow duplication mode with a basic dual-nozzle extruder
//#define DUAL_NOZZLE_DUPLICATION_MODE

// By default pololu step drivers require an active high signal. However, some high power drivers require
an active low signal as step.
#define INVERT_X_STEP_PIN false
#define INVERT_Y_STEP_PIN false
#define INVERT_Z_STEP_PIN false
#define INVERT_E_STEP_PIN false

// Default stepper release if idle. Set to 0 to deactivate.
// Steppers will shut down DEFAULT_STEPPER_DEACTIVE_TIME seconds after the last move when
DISABLE_INACTIVE_? is true.
// Time can be set by M18 and M84.
#define DEFAULT_STEPPER_DEACTIVE_TIME 120
#define DISABLE_INACTIVE_X true
#define DISABLE_INACTIVE_Y true
#define DISABLE_INACTIVE_Z true  // set to false if the nozzle will fall down on your printed part when
print has finished.
#define DISABLE_INACTIVE_E true

#define DEFAULT_MINIMUMFEEDRATE       0.0     // minimum feedrate
#define DEFAULT_MINTRAVELFEEDRATE     0.0

//#define HOME_AFTER_DEACTIVATE  // Require rehoming after steppers are deactivated

// @section lcd
```

```
#if ENABLED(ULTIPANEL)
  #define MANUAL_FEEDRATE {50*60, 50*60, 4*60, 60} // Feedrates for manual moves along X, Y, Z, E from
panel
  #define ULTIPANEL_FEEDMULTIPLY  // Comment to disable setting feedrate multiplier via encoder
#endif

// @section extras

// minimum time in microseconds that a movement needs to take if the buffer is emptied.
#define DEFAULT_MINSEGMENTTIME        20000

// If defined the movements slow down when the look ahead buffer is only half full
#define SLOWDOWN

// Frequency limit
// See nophead's blog for more info
// Not working O
//#define XY_FREQUENCY_LIMIT  15

// Minimum planner junction speed. Sets the default minimum speed the planner plans for at the end
// of the buffer and all stops. This should not be much greater than zero and should only be changed
// if unwanted behavior is observed on a user's machine when running at very slow speeds.
#define MINIMUM_PLANNER_SPEED 0.05 // (mm/sec)

// Microstep setting (Only functional when stepper driver microstep pins are connected to MCU.
#define MICROSTEP_MODES {16,16,16,16,16} // [1,2,4,8,16]

/**
 *  @section  stepper motor current
 *
 *  Some boards have a means of setting the stepper motor current via firmware.
 *
 *  The power on motor currents are set by:
 *    PWM_MOTOR_CURRENT - used by MINIRAMBO & ULTIMAIN_2
 *                        known compatible chips: A4982
 *    DIGIPOT_MOTOR_CURRENT - used by BQ_ZUM_MEGA_3D, RAMBO & SCOOVO_X9H
 *                        known compatible chips: AD5206
 *    DAC_MOTOR_CURRENT_DEFAULT - used by PRINTRBOARD_REVF & RIGIDBOARD_V2
 *                        known compatible chips: MCP4728
 *    DIGIPOT_I2C_MOTOR_CURRENTS - used by 5DPRINT, AZTEEG_X3_PRO, MIGHTYBOARD_REVE
 *                        known compatible chips: MCP4451, MCP4018
 *
 *  Motor currents can also be set by M907 - M910 and by the LCD.
 *    M907 - applies to all.
 *    M908 - BQ_ZUM_MEGA_3D, RAMBO, PRINTRBOARD_REVF, RIGIDBOARD_V2 & SCOOVO_X9H
 *    M909, M910 & LCD - only PRINTRBOARD_REVF & RIGIDBOARD_V2
 */
//#define PWM_MOTOR_CURRENT { 1300, 1300, 1250 }          // Values in milliamps
//#define DIGIPOT_MOTOR_CURRENT { 135,135,135,135,135 }   // Values 0-255 (RAMBO 135 = ~0.75A, 185 = ~1A)
//#define DAC_MOTOR_CURRENT_DEFAULT { 70, 80, 90, 80 }    // Default drive percent - X, Y, Z, E axis

// Uncomment to enable an I2C based DIGIPOT like on the Azteeg X3 Pro
//#define DIGIPOT_I2C
//#define DIGIPOT_MCP4018          // Requires library from https://github.com/stawel/SlowSoftI2CMaster
#define DIGIPOT_I2C_NUM_CHANNELS 8 // 5DPRINT: 4     AZTEEG_X3_PRO: 8
// Actual motor currents in Amps, need as many here as DIGIPOT_I2C_NUM_CHANNELS
#define DIGIPOT_I2C_MOTOR_CURRENTS { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 }  //  AZTEEG_X3_PRO

//===========================================================================
//=============================Additional Features===========================
//===========================================================================

#define ENCODER_RATE_MULTIPLIER         // If defined, certain menu edit operations automatically
multiply the steps when the encoder is moved quickly
#define ENCODER_10X_STEPS_PER_SEC 75    // If the encoder steps per sec exceeds this value, multiply
steps moved x10 to quickly advance the value
#define ENCODER_100X_STEPS_PER_SEC 160  // If the encoder steps per sec exceeds this value, multiply
steps moved x100 to really quickly advance the value

//#define CHDK 4        //Pin for triggering CHDK to take a picture see how to use it here http://
captain-slow.dk/2014/03/09/3d-printing-timelapses/
#define CHDK_DELAY 50 //How long in ms the pin should stay HIGH before going LOW again

// @section lcd

// Include a page of printer information in the LCD Main Menu
//#define LCD_INFO_MENU

// Scroll a longer status message into view
//#define STATUS_MESSAGE_SCROLLING
```

```
// On the Info Screen, display XY with one decimal place when possible
//#define LCD_DECIMAL_SMALL_XY

// The timeout (in ms) to return to the status screen from sub-menus
//#define LCD_TIMEOUT_TO_STATUS 15000

#if ENABLED(SDSUPPORT)

  // Some RAMPS and other boards don't detect when an SD card is inserted. You can work
  // around this by connecting a push button or single throw switch to the pin defined
  // as SD_DETECT_PIN in your board's pins definitions.
  // This setting should be disabled unless you are using a push button, pulling the pin to ground.
  // Note: This is always disabled for ULTIPANEL (except ELB_FULL_GRAPHIC_CONTROLLER).
  #define SD_DETECT_INVERTED

  #define SD_FINISHED_STEPPERRELEASE true  //if sd support and the file is finished: disable steppers?
  #define SD_FINISHED_RELEASECOMMAND "M84 X Y Z E" // You might want to keep the z enabled so your bed
stays in place.

  #define SDCARD_RATHERRECENTFIRST  //reverse file order of sd card menu display. Its sorted practically
after the file system block order.
  // if a file is deleted, it frees a block. hence, the order is not purely chronological. To still have
auto0.g accessible, there is again the option to do that.
  // using:
  //#define MENU_ADDAUTOSTART

  /**
   * Sort SD file listings in alphabetical order.
   *
   * With this option enabled, items on SD cards will be sorted
   * by name for easier navigation.
   *
   * By default...
   *
   *  - Use the slowest -but safest- method for sorting.
   *  - Folders are sorted to the top.
   *  - The sort key is statically allocated.
   *  - No added G-code (M34) support.
   *  - 40 item sorting limit. (Items after the first 40 are unsorted.)
   *
   * SD sorting uses static allocation (as set by SDSORT_LIMIT), allowing the
   * compiler to calculate the worst-case usage and throw an error if the SRAM
   * limit is exceeded.
   *
   *  - SDSORT_USES_RAM provides faster sorting via a static directory buffer.
   *  - SDSORT_USES_STACK does the same, but uses a local stack-based buffer.
   *  - SDSORT_CACHE_NAMES will retain the sorted file listing in RAM. (Expensive!)
   *  - SDSORT_DYNAMIC_RAM only uses RAM when the SD menu is visible. (Use with caution!)
   */
  //#define SDCARD_SORT_ALPHA

  // SD Card Sorting options
  #if ENABLED(SDCARD_SORT_ALPHA)
    #define SDSORT_LIMIT       40     // Maximum number of sorted items (10-256).
    #define FOLDER_SORTING     -1     // -1=above  0=none  1=below
    #define SDSORT_GCODE       false  // Allow turning sorting on/off with LCD and M34 g-code.
    #define SDSORT_USES_RAM    false  // Pre-allocate a static array for faster pre-sorting.
    #define SDSORT_USES_STACK  false  // Prefer the stack for pre-sorting to give back some SRAM.
(Negated by next 2 options.)
    #define SDSORT_CACHE_NAMES false  // Keep sorted items in RAM longer for speedy performance. Most
expensive option.
    #define SDSORT_DYNAMIC_RAM false  // Use dynamic allocation (within SD menus). Least expensive
option. Set SDSORT_LIMIT before use!
  #endif

  // Show a progress bar on HD44780 LCDs for SD printing
  //#define LCD_PROGRESS_BAR

  #if ENABLED(LCD_PROGRESS_BAR)
    // Amount of time (ms) to show the bar
    #define PROGRESS_BAR_BAR_TIME 2000
    // Amount of time (ms) to show the status message
    #define PROGRESS_BAR_MSG_TIME 3000
    // Amount of time (ms) to retain the status message (0=forever)
    #define PROGRESS_MSG_EXPIRE   0
    // Enable this to show messages for MSG_TIME then hide them
    //#define PROGRESS_MSG_ONCE
    // Add a menu item to test the progress bar:
    //#define LCD_PROGRESS_BAR_TEST
  #endif
```

```
  // This allows hosts to request long names for files and folders with M33
  //#define LONG_FILENAME_HOST_SUPPORT

  // This option allows you to abort SD printing when any endstop is triggered.
  // This feature must be enabled with "M540 S1" or from the LCD menu.
  // To have any effect, endstops must be enabled during SD printing.
  //#define ABORT_ON_ENDSTOP_HIT_FEATURE_ENABLED

#endif // SDSUPPORT

/**
 * Additional options for Graphical Displays
 *
 * Use the optimizations here to improve printing performance,
 * which can be adversely affected by graphical display drawing,
 * especially when doing several short moves, and when printing
 * on DELTA and SCARA machines.
 *
 * Some of these options may result in the display lagging behind
 * controller events, as there is a trade-off between reliable
 * printing performance versus fast display updates.
 */
#if ENABLED(DOGLCD)
  // Enable to save many cycles by drawing a hollow frame on the Info Screen
  #define XYZ_HOLLOW_FRAME
  // Enable to save many cycles by drawing a hollow frame on Menu Screens
  #define MENU_HOLLOW_FRAME

  // A bigger font is available for edit items. Costs 3120 bytes of PROGMEM.
  // Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.
  //#define USE_BIG_EDIT_FONT

  // A smaller font may be used on the Info Screen. Costs 2300 bytes of PROGMEM.
  // Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.
  //#define USE_SMALL_INFOFONT

  // Enable this option and reduce the value to optimize screen updates.
  // The normal delay is 10µs. Use the lowest value that still gives a reliable display.
  //#define DOGM_SPI_DELAY_US 5
#endif // DOGLCD

// @section safety

// The hardware watchdog should reset the microcontroller disabling all outputs,
// in case the firmware gets stuck and doesn't do temperature regulation.
#define USE_WATCHDOG

#if ENABLED(USE_WATCHDOG)
  // If you have a watchdog reboot in an ArduinoMega2560 then the device will hang forever, as a watchdog reset will leave the watchdog on.
  // The "WATCHDOG_RESET_MANUAL" goes around this by not using the hardware reset.
  //  However, THIS FEATURE IS UNSAFE!, as it will only work if interrupts are disabled. And the code could hang in an interrupt routine with interrupts disabled.
  //#define WATCHDOG_RESET_MANUAL
#endif

// @section lcd

/**
 * Babystepping enables movement of the axes by tiny increments without changing
 * the current position values. This feature is used primarily to adjust the Z
 * axis in the first layer of a print in real-time.
 *
 * Warning: Does not respect endstops!
 */
#define BABYSTEPPING
#if ENABLED(BABYSTEPPING)
  //#define BABYSTEP_XY              // Also enable X/Y Babystepping. Not supported on DELTA!
  #define BABYSTEP_INVERT_Z false  // Change if Z babysteps should go the other way
  #define BABYSTEP_MULTIPLICATOR 2 // Babysteps are very small. Increase for faster motion.
  //#define BABYSTEP_ZPROBE_OFFSET // Enable to combine M851 and Babystepping
  #define DOUBLECLICK_FOR_Z_BABYSTEPPING // Double-click on the Status Screen for Z Babystepping.
  #define DOUBLECLICK_MAX_INTERVAL 1250 // Maximum interval between clicks, in milliseconds.
                                        // Note: Extra time may be added to mitigate controller latency.
#endif

// @section extruder

// extruder advance constant (s2/mm3)
//
```

```
// advance (steps) = STEPS_PER_CUBIC_MM_E * EXTRUDER_ADVANCE_K * cubic mm per second ^ 2
//
// Hooke's law says:    force = k * distance
// Bernoulli's principle says:  v ^ 2 / 2 + g . h + pressure / density = constant
// so: v ^ 2 is proportional to number of steps we advance the extruder
//#define ADVANCE

#if ENABLED(ADVANCE)
  #define EXTRUDER_ADVANCE_K .0
  #define D_FILAMENT 2.85
#endif

/**
 * Implementation of linear pressure control
 *
 * Assumption: advance = k * (delta velocity)
 * K=0 means advance disabled.
 * See Marlin documentation for calibration instructions.
 */
//#define LIN_ADVANCE

#if ENABLED(LIN_ADVANCE)
  #define LIN_ADVANCE_K 75

  /**
   * Some Slicers produce Gcode with randomly jumping extrusion widths occasionally.
   * For example within a 0.4mm perimeter it may produce a single segment of 0.05mm width.
   * While this is harmless for normal printing (the fluid nature of the filament will
   * close this very, very tiny gap), it throws off the LIN_ADVANCE pressure adaption.
   *
   * For this case LIN_ADVANCE_E_D_RATIO can be used to set the extrusion:distance ratio
   * to a fixed value. Note that using a fixed ratio will lead to wrong nozzle pressures
   * if the slicer is using variable widths or layer heights within one print!
   *
   * This option sets the default E:D ratio at startup. Use `M900` to override this value.
   *
   * Example: `M900 W0.4 H0.2 D1.75`, where:
   *   - W is the extrusion width in mm
   *   - H is the layer height in mm
   *   - D is the filament diameter in mm
   *
   * Example: `M900 R0.0458` to set the ratio directly.
   *
   * Set to 0 to auto-detect the ratio based on given Gcode G1 print moves.
   *
   * Slic3r (including Průša Control) produces Gcode compatible with the automatic mode.
   * Cura (as of this writing) may produce Gcode incompatible with the automatic mode.
   */
  #define LIN_ADVANCE_E_D_RATIO 0 // The calculated ratio (or 0) according to the formula W * H / ((D /
2) ^ 2 * PI)
                                  // Example: 0.4 * 0.2 / ((1.75 / 2) ^ 2 * PI) = 0.033260135
#endif

// @section leveling

// Default mesh area is an area with an inset margin on the print area.
// Below are the macros that are used to define the borders for the mesh area,
// made available here for specialized needs, ie dual extruder setup.
#if ENABLED(MESH_BED_LEVELING)
  #define MESH_MIN_X MESH_INSET
  #define MESH_MAX_X (X_BED_SIZE - (MESH_INSET))
  #define MESH_MIN_Y MESH_INSET
  #define MESH_MAX_Y (Y_BED_SIZE - (MESH_INSET))
#elif ENABLED(AUTO_BED_LEVELING_UBL)
  #define UBL_MESH_MIN_X UBL_MESH_INSET
  #define UBL_MESH_MAX_X (X_BED_SIZE - (UBL_MESH_INSET))
  #define UBL_MESH_MIN_Y UBL_MESH_INSET
  #define UBL_MESH_MAX_Y (Y_BED_SIZE - (UBL_MESH_INSET))

  // If this is defined, the currently active mesh will be saved in the
  // current slot on M500.
  #define UBL_SAVE_ACTIVE_ON_M500
#endif

// @section extras

//
// G2/G3 Arc Support
//
#define ARC_SUPPORT               // Disable this feature to save ~3226 bytes
#if ENABLED(ARC_SUPPORT)
```

```
  #define MM_PER_ARC_SEGMENT  1   // Length of each arc segment
  #define N_ARC_CORRECTION   25   // Number of intertpolated segments between corrections
  //#define ARC_P_CIRCLES         // Enable the 'P' parameter to specify complete circles
  //#define CNC_WORKSPACE_PLANES  // Allow G2/G3 to operate in XY, ZX, or YZ planes
#endif

// Support for G5 with XYZE destination and IJPQ offsets. Requires ~2666 bytes.
//#define BEZIER_CURVE_SUPPORT

// G38.2 and G38.3 Probe Target
// Enable PROBE_DOUBLE_TOUCH if you want G38 to double touch
//#define G38_PROBE_TARGET
#if ENABLED(G38_PROBE_TARGET)
  #define G38_MINIMUM_MOVE 0.0275 // minimum distance in mm that will produce a move (determined using
the print statement in check_move)
#endif

// Moves (or segments) with fewer steps than this will be joined with the next move
#define MIN_STEPS_PER_SEGMENT 6

// The minimum pulse width (in µs) for stepping a stepper.
// Set this if you find stepping unreliable, or if using a very fast CPU.
#define MINIMUM_STEPPER_PULSE 0 // (µs) The smallest stepper pulse allowed

// @section temperature

// Control heater 0 and heater 1 in parallel.
//#define HEATERS_PARALLEL

//===========================================================================
//=============================== Buffers ===================================
//===========================================================================

// @section hidden

// The number of linear motions that can be in the plan at any give time.
// THE BLOCK_BUFFER_SIZE NEEDS TO BE A POWER OF 2, i.g. 8,16,32 because shifts and ors are used to do the
ring-buffering.
#if ENABLED(SDSUPPORT)
  #define BLOCK_BUFFER_SIZE 16 // SD,LCD,Buttons take more memory, block buffer needs to be smaller
#else
  #define BLOCK_BUFFER_SIZE 16 // maximize block buffer
#endif

// @section serial

// The ASCII buffer for serial input
#define MAX_CMD_SIZE 96
#define BUFSIZE 4

// Transfer Buffer Size
// To save 386 bytes of PROGMEM (and TX_BUFFER_SIZE+3 bytes of RAM) set to 0.
// To buffer a simple "ok" you need 4 bytes.
// For ADVANCED_OK (M105) you need 32 bytes.
// For debug-echo: 128 bytes for the optimal speed.
// Other output doesn't need to be that speedy.
// :[0, 2, 4, 8, 16, 32, 64, 128, 256]
#define TX_BUFFER_SIZE 0

// Enable an emergency-command parser to intercept certain commands as they
// enter the serial receive buffer, so they cannot be blocked.
// Currently handles M108, M112, M410
// Does not work on boards using AT90USB (USBCON) processors!
//#define EMERGENCY_PARSER

// Bad Serial-connections can miss a received command by sending an 'ok'
// Therefore some clients abort after 30 seconds in a timeout.
// Some other clients start sending commands while receiving a 'wait'.
// This "wait" is only sent when the buffer is empty. 1 second is a good value here.
//#define NO_TIMEOUTS 1000 // Milliseconds

// Some clients will have this feature soon. This could make the NO_TIMEOUTS unnecessary.
//#define ADVANCED_OK

// @section extras

/**
 * Firmware-based and LCD-controlled retract
 *
 * Add G10 / G11 commands for automatic firmware-based retract / recover.
 * Use M207 and M208 to define parameters for retract / recover.
```

```
 *
 * Use M209 to enable or disable auto-retract.
 * With auto-retract enabled, all G1 E moves within the set range
 * will be converted to firmware-based retract/recover moves.
 *
 * Be sure to turn off auto-retract during filament change.
 *
 * Note that M207 / M208 / M209 settings are saved to EEPROM.
 *
 */
//#define FWRETRACT  // ONLY PARTIALLY TESTED
#if ENABLED(FWRETRACT)
  #define MIN_AUTORETRACT 0.1             // When auto-retract is on, convert E moves of this length and
over
  #define MAX_AUTORETRACT 10.0            // Upper limit for auto-retract conversion
  #define RETRACT_LENGTH 3                // Default retract length (positive mm)
  #define RETRACT_LENGTH_SWAP 13          // Default swap retract length (positive mm), for extruder
change
  #define RETRACT_FEEDRATE 45             // Default feedrate for retracting (mm/s)
  #define RETRACT_ZLIFT 0                 // Default retract Z-lift
  #define RETRACT_RECOVER_LENGTH 0        // Default additional recover length (mm, added to retract
length when recovering)
  #define RETRACT_RECOVER_LENGTH_SWAP 0   // Default additional swap recover length (mm, added to retract
length when recovering from extruder change)
  #define RETRACT_RECOVER_FEEDRATE 8      // Default feedrate for recovering from retraction (mm/s)
  #define RETRACT_RECOVER_FEEDRATE_SWAP 8 // Default feedrate for recovering from swap retraction (mm/s)
#endif

/**
 * Advanced Pause
 * Experimental feature for filament change support and for parking the nozzle when paused.
 * Adds the GCode M600 for initiating filament change.
 * If PARK_HEAD_ON_PAUSE enabled, adds the GCode M125 to pause printing and park the nozzle.
 *
 * Requires an LCD display.
 * This feature is required for the default FILAMENT_RUNOUT_SCRIPT.
 */
//#define ADVANCED_PAUSE_FEATURE
#if ENABLED(ADVANCED_PAUSE_FEATURE)
  #define PAUSE_PARK_X_POS 3                  // X position of hotend
  #define PAUSE_PARK_Y_POS 3                  // Y position of hotend
  #define PAUSE_PARK_Z_ADD 10                 // Z addition of hotend (lift)
  #define PAUSE_PARK_XY_FEEDRATE 100          // X and Y axes feedrate in mm/s (also used for delta
printers Z axis)
  #define PAUSE_PARK_Z_FEEDRATE 5             // Z axis feedrate in mm/s (not used for delta printers)
  #define PAUSE_PARK_RETRACT_FEEDRATE 60      // Initial retract feedrate in mm/s
  #define PAUSE_PARK_RETRACT_LENGTH 2         // Initial retract in mm
                                              // It is a short retract used immediately after print
interrupt before move to filament exchange position
  #define FILAMENT_CHANGE_UNLOAD_FEEDRATE 10  // Unload filament feedrate in mm/s - filament unloading
can be fast
  #define FILAMENT_CHANGE_UNLOAD_LENGTH 100   // Unload filament length from hotend in mm
                                              // Longer length for bowden printers to unload filament
from whole bowden tube,
                                              // shorter length for printers without bowden to unload
filament from extruder only,
                                              // 0 to disable unloading for manual unloading
  #define FILAMENT_CHANGE_LOAD_FEEDRATE 6     // Load filament feedrate in mm/s - filament loading into
the bowden tube can be fast
  #define FILAMENT_CHANGE_LOAD_LENGTH 0       // Load filament length over hotend in mm
                                              // Longer length for bowden printers to fast load filament
into whole bowden tube over the hotend,
                                              // Short or zero length for printers without bowden where
loading is not used
  #define ADVANCED_PAUSE_EXTRUDE_FEEDRATE 3   // Extrude filament feedrate in mm/s - must be slower than
load feedrate
  #define ADVANCED_PAUSE_EXTRUDE_LENGTH 50    // Extrude filament length in mm after filament is loaded
over the hotend,
                                              // 0 to disable for manual extrusion
                                              // Filament can be extruded repeatedly from the filament
exchange menu to fill the hotend,
                                              // or until outcoming filament color is not clear for
filament color change
  #define PAUSE_PARK_NOZZLE_TIMEOUT 45        // Turn off nozzle if user doesn't change filament within
this time limit in seconds
  #define FILAMENT_CHANGE_NUMBER_OF_ALERT_BEEPS 5 // Number of alert beeps before printer goes quiet
  #define PAUSE_PARK_NO_STEPPER_TIMEOUT       // Enable to have stepper motors hold position during
filament change
                                              // even if it takes longer than
DEFAULT_STEPPER_DEACTIVE_TIME.
  //#define PARK_HEAD_ON_PAUSE                // Go to filament change position on pause, return to print
```

```
position on resume
  //#define HOME_BEFORE_FILAMENT_CHANGE       // Ensure homing has been completed prior to parking for
filament change
#endif

// @section tmc

/**
 * Enable this section if you have TMC26X motor drivers.
 * You will need to import the TMC26XStepper library into the Arduino IDE for this
 * (https://github.com/trinamic/TMC26XStepper.git)
 */
//#define HAVE_TMCDRIVER

#if ENABLED(HAVE_TMCDRIVER)

  //#define X_IS_TMC
  //#define X2_IS_TMC
  //#define Y_IS_TMC
  //#define Y2_IS_TMC
  //#define Z_IS_TMC
  //#define Z2_IS_TMC
  //#define E0_IS_TMC
  //#define E1_IS_TMC
  //#define E2_IS_TMC
  //#define E3_IS_TMC
  //#define E4_IS_TMC

  #define X_MAX_CURRENT     1000 // in mA
  #define X_SENSE_RESISTOR    91 // in mOhms
  #define X_MICROSTEPS        16 // number of microsteps

  #define X2_MAX_CURRENT    1000
  #define X2_SENSE_RESISTOR   91
  #define X2_MICROSTEPS       16

  #define Y_MAX_CURRENT     1000
  #define Y_SENSE_RESISTOR    91
  #define Y_MICROSTEPS        16

  #define Y2_MAX_CURRENT    1000
  #define Y2_SENSE_RESISTOR   91
  #define Y2_MICROSTEPS       16

  #define Z_MAX_CURRENT     1000
  #define Z_SENSE_RESISTOR    91
  #define Z_MICROSTEPS        16

  #define Z2_MAX_CURRENT    1000
  #define Z2_SENSE_RESISTOR   91
  #define Z2_MICROSTEPS       16

  #define E0_MAX_CURRENT    1000
  #define E0_SENSE_RESISTOR   91
  #define E0_MICROSTEPS       16

  #define E1_MAX_CURRENT    1000
  #define E1_SENSE_RESISTOR   91
  #define E1_MICROSTEPS       16

  #define E2_MAX_CURRENT    1000
  #define E2_SENSE_RESISTOR   91
  #define E2_MICROSTEPS       16

  #define E3_MAX_CURRENT    1000
  #define E3_SENSE_RESISTOR   91
  #define E3_MICROSTEPS       16

  #define E4_MAX_CURRENT    1000
  #define E4_SENSE_RESISTOR   91
  #define E4_MICROSTEPS       16

#endif

// @section TMC2130

/**
 * Enable this for SilentStepStick Trinamic TMC2130 SPI-configurable stepper drivers.
 *
 * You'll also need the TMC2130Stepper Arduino library
 * (https://github.com/teemuatlut/TMC2130Stepper).
```

```
 *
 * To use TMC2130 stepper drivers in SPI mode connect your SPI2130 pins to
 * the hardware SPI interface on your board and define the required CS pins
 * in your `pins_MYBOARD.h` file. (e.g., RAMPS 1.4 uses AUX3 pins `X_CS_PIN 53`, `Y_CS_PIN 49`, etc.).
 */
#define HAVE_TMC2130

#if ENABLED(HAVE_TMC2130)

  // CHOOSE YOUR MOTORS HERE, THIS IS MANDATORY
  #define X_IS_TMC2130
  //#define X2_IS_TMC2130
  #define Y_IS_TMC2130
  //#define Y2_IS_TMC2130
  //#define Z_IS_TMC2130
  //#define Z2_IS_TMC2130
  //#define E0_IS_TMC2130
  //#define E1_IS_TMC2130
  //#define E2_IS_TMC2130
  //#define E3_IS_TMC2130
  //#define E4_IS_TMC2130

  /**
   * Stepper driver settings
   */

#define R_SENSE           0.11  // R_sense resistor for SilentStepStick2130
#define HOLD_MULTIPLIER    0.5  // Scales down the holding current from run current
#define INTERPOLATE          0  // Interpolate X/Y/Z_MICROSTEPS to 256

#define X_CURRENT         200  // rms current in mA. Multiply by 1.41 for peak current.
#define X_MICROSTEPS       16  // 0..256

#define Y_CURRENT         200
#define Y_MICROSTEPS       16

#define Z_CURRENT         200
#define Z_MICROSTEPS       16

//#define X2_CURRENT      1000
//#define X2_MICROSTEPS     16

//#define Y2_CURRENT      1000
//#define Y2_MICROSTEPS     16

//#define Z2_CURRENT      1000
//#define Z2_MICROSTEPS     16

//#define E0_CURRENT      1000
//#define E0_MICROSTEPS     16

//#define E1_CURRENT      1000
//#define E1_MICROSTEPS     16

//#define E2_CURRENT      1000
//#define E2_MICROSTEPS     16

  //#define E3_CURRENT      1000
  //#define E3_MICROSTEPS     16

  //#define E4_CURRENT      1000
  //#define E4_MICROSTEPS     16

  /**
   * Use Trinamic's ultra quiet stepping mode.
   * When disabled, Marlin will use spreadCycle stepping mode.
   */
  //#define STEALTHCHOP

  /**
   * Let Marlin automatically control stepper current.
   * This is still an experimental feature.
   * Increase current every 5s by CURRENT_STEP until stepper temperature prewarn gets triggered,
   * then decrease current by CURRENT_STEP until temperature prewarn is cleared.
   * Adjusting starts from X/Y/Z/E_CURRENT but will not increase over AUTO_ADJUST_MAX
   * Relevant g-codes:
   * M906 - Set or get motor current in milliamps using axis codes X, Y, Z, E. Report values if no axis
codes given.
   * M906 S1 - Start adjusting current
   * M906 S0 - Stop adjusting current
   * M911 - Report stepper driver overtemperature pre-warn condition.
```

```
 * M912 - Clear stepper driver overtemperature pre-warn condition flag.
 */
//#define AUTOMATIC_CURRENT_CONTROL

#if ENABLED(AUTOMATIC_CURRENT_CONTROL)
  #define CURRENT_STEP          50  // [mA]
  #define AUTO_ADJUST_MAX     1300  // [mA], 1300mA_rms = 1840mA_peak
  #define REPORT_CURRENT_CHANGE
#endif

/**
 * The driver will switch to spreadCycle when stepper speed is over HYBRID_THRESHOLD.
 * This mode allows for faster movements at the expense of higher noise levels.
 * STEALTHCHOP needs to be enabled.
 * M913 X/Y/Z/E to live tune the setting
 */
//#define HYBRID_THRESHOLD

#define X_HYBRID_THRESHOLD     100  // [mm/s]
#define X2_HYBRID_THRESHOLD    100
#define Y_HYBRID_THRESHOLD     100
#define Y2_HYBRID_THRESHOLD    100
#define Z_HYBRID_THRESHOLD       4
#define Z2_HYBRID_THRESHOLD      4
#define E0_HYBRID_THRESHOLD     30
#define E1_HYBRID_THRESHOLD     30
#define E2_HYBRID_THRESHOLD     30
#define E3_HYBRID_THRESHOLD     30
#define E4_HYBRID_THRESHOLD     30

/**
 * Use stallGuard2 to sense an obstacle and trigger an endstop.
 * You need to place a wire from the driver's DIAG1 pin to the X/Y endstop pin.
 * If used along with STEALTHCHOP, the movement will be louder when homing. This is normal.
 *
 * X/Y_HOMING_SENSITIVITY is used for tuning the trigger sensitivity.
 * Higher values make the system LESS sensitive.
 * Lower value make the system MORE sensitive.
 * Too low values can lead to false positives, while too high values will collide the axis without
triggering.
 * It is advised to set X/Y_HOME_BUMP_MM to 0.
 * M914 X/Y to live tune the setting
 */
//#define SENSORLESS_HOMING

#if ENABLED(SENSORLESS_HOMING)
  #define X_HOMING_SENSITIVITY  19
  #define Y_HOMING_SENSITIVITY  19
#endif

/**
 * You can set your own advanced settings by filling in predefined functions.
 * A list of available functions can be found on the library github page
 * https://github.com/teemuatlut/TMC2130Stepper
 *
 * Example:
 * #define TMC2130_ADV() { \
 *   stepperX.diag0_temp_prewarn(1); \
 *   stepperX.interpolate(0); \
 * }
 */
#define  TMC2130_ADV() {  }

#endif // HAVE_TMC2130

// @section L6470

/**
 * Enable this section if you have L6470 motor drivers.
 * You need to import the L6470 library into the Arduino IDE for this.
 * (https://github.com/ameyer/Arduino-L6470)
 */

//#define HAVE_L6470DRIVER
#if ENABLED(HAVE_L6470DRIVER)

  //#define X_IS_L6470
  //#define X2_IS_L6470
  //#define Y_IS_L6470
  //#define Y2_IS_L6470
  //#define Z_IS_L6470
```

```
  //#define Z2_IS_L6470
  //#define E0_IS_L6470
  //#define E1_IS_L6470
  //#define E2_IS_L6470
  //#define E3_IS_L6470
  //#define E4_IS_L6470

  #define X_MICROSTEPS     16 // number of microsteps
  #define X_K_VAL          50 // 0 - 255, Higher values, are higher power. Be careful not to go too high
  #define X_OVERCURRENT   2000 // maxc current in mA. If the current goes over this value, the driver
will switch off
  #define X_STALLCURRENT  1500 // current in mA where the driver will detect a stall

  #define X2_MICROSTEPS     16
  #define X2_K_VAL          50
  #define X2_OVERCURRENT  2000
  #define X2_STALLCURRENT 1500

  #define Y_MICROSTEPS      16
  #define Y_K_VAL           50
  #define Y_OVERCURRENT   2000
  #define Y_STALLCURRENT  1500

  #define Y2_MICROSTEPS     16
  #define Y2_K_VAL          50
  #define Y2_OVERCURRENT  2000
  #define Y2_STALLCURRENT 1500

  #define Z_MICROSTEPS      16
  #define Z_K_VAL           50
  #define Z_OVERCURRENT   2000
  #define Z_STALLCURRENT  1500

  #define Z2_MICROSTEPS     16
  #define Z2_K_VAL          50
  #define Z2_OVERCURRENT  2000
  #define Z2_STALLCURRENT 1500

  #define E0_MICROSTEPS     16
  #define E0_K_VAL          50
  #define E0_OVERCURRENT  2000
  #define E0_STALLCURRENT 1500

  #define E1_MICROSTEPS     16
  #define E1_K_VAL          50
  #define E1_OVERCURRENT  2000
  #define E1_STALLCURRENT 1500

  #define E2_MICROSTEPS     16
  #define E2_K_VAL          50
  #define E2_OVERCURRENT  2000
  #define E2_STALLCURRENT 1500

  #define E3_MICROSTEPS     16
  #define E3_K_VAL          50
  #define E3_OVERCURRENT  2000
  #define E3_STALLCURRENT 1500

  #define E4_MICROSTEPS     16
  #define E4_K_VAL          50
  #define E4_OVERCURRENT  2000
  #define E4_STALLCURRENT 1500

#endif

/**
 * TWI/I2C BUS
 *
 * This feature is an EXPERIMENTAL feature so it shall not be used on production
 * machines. Enabling this will allow you to send and receive I2C data from slave
 * devices on the bus.
 *
 * ; Example #1
 * ; This macro send the string "Marlin" to the slave device with address 0x63 (99)
 * ; It uses multiple M260 commands with one B<base 10> arg
 * M260 A99  ; Target slave address
 * M260 B77  ; M
 * M260 B97  ; a
 * M260 B114 ; r
 * M260 B108 ; l
 * M260 B105 ; i
```

```
 * M260 B110 ; n
 * M260 S1   ; Send the current buffer
 *
 * ; Example #2
 * ; Request 6 bytes from slave device with address 0x63 (99)
 * M261 A99 B5
 *
 * ; Example #3
 * ; Example serial output of a M261 request
 * echo:i2c-reply: from:99 bytes:5 data:hello
 */

// @section i2cbus

//#define EXPERIMENTAL_I2CBUS
#define I2C_SLAVE_ADDRESS  0 // Set a value from 8 to 127 to act as a slave

// @section extras

/**
 * Spindle & Laser control
 *
 * Add the M3, M4, and M5 commands to turn the spindle/laser on and off, and
 * to set spindle speed, spindle direction, and laser power.
 *
 * SuperPid is a router/spindle speed controller used in the CNC milling community.
 * Marlin can be used to turn the spindle on and off. It can also be used to set
 * the spindle speed from 5,000 to 30,000 RPM.
 *
 * You'll need to select a pin for the ON/OFF function and optionally choose a 0-5V
 * hardware PWM pin for the speed control and a pin for the rotation direction.
 *
 * See http://marlinfw.org/docs/configuration/laser_spindle.html for more config details.
 */
//#define SPINDLE_LASER_ENABLE
#if ENABLED(SPINDLE_LASER_ENABLE)

  #define SPINDLE_LASER_ENABLE_INVERT   false  // set to "true" if the on/off function is reversed
  #define SPINDLE_LASER_PWM             true   // set to true if your controller supports setting the
speed/power
  #define SPINDLE_LASER_PWM_INVERT      true   // set to "true" if the speed/power goes up when you want
it to go slower
  #define SPINDLE_LASER_POWERUP_DELAY   5000   // delay in milliseconds to allow the spindle/laser to
come up to speed/power
  #define SPINDLE_LASER_POWERDOWN_DELAY 5000   // delay in milliseconds to allow the spindle to stop
  #define SPINDLE_DIR_CHANGE            true   // set to true if your spindle controller supports
changing spindle direction
  #define SPINDLE_INVERT_DIR            false
  #define SPINDLE_STOP_ON_DIR_CHANGE    true   // set to true if Marlin should stop the spindle before
changing rotation direction

  /**
   *  The M3 & M4 commands use the following equation to convert PWM duty cycle to speed/power
   *
   *  SPEED/POWER = PWM duty cycle * SPEED_POWER_SLOPE + SPEED_POWER_INTERCEPT
   *    where PWM duty cycle varies from 0 to 255
   *
   *  set the following for your controller (ALL MUST BE SET)
   */

  #define SPEED_POWER_SLOPE    118.4
  #define SPEED_POWER_INTERCEPT  0
  #define SPEED_POWER_MIN     5000
  #define SPEED_POWER_MAX    30000    // SuperPID router controller 0 - 30,000 RPM

  //#define SPEED_POWER_SLOPE      0.3922
  //#define SPEED_POWER_INTERCEPT  0
  //#define SPEED_POWER_MIN       10
  //#define SPEED_POWER_MAX      100      // 0-100%
#endif

/**
 * M43 - display pin status, watch pins for changes, watch endstops & toggle LED, Z servo probe test,
toggle pins
 */
//#define PINS_DEBUGGING

/**
 * Auto-report temperatures with M155 S<seconds>
 */
#define AUTO_REPORT_TEMPERATURES
```

```
/**
 * Include capabilities in M115 output
 */
#define EXTENDED_CAPABILITIES_REPORT

/**
 * Volumetric extrusion default state
 * Activate to make volumetric extrusion the default method,
 * with DEFAULT_NOMINAL_FILAMENT_DIA as the default diameter.
 *
 * M200 D0 to disable, M200 Dn to set a new diameter.
 */
//#define VOLUMETRIC_DEFAULT_ON

/**
 * Enable this option for a leaner build of Marlin that removes all
 * workspace offsets, simplifying coordinate transformations, leveling, etc.
 *
 *  - M206 and M428 are disabled.
 *  - G92 will revert to its behavior from Marlin 1.0.
 */
//#define NO_WORKSPACE_OFFSETS

/**
 * Set the number of proportional font spaces required to fill up a typical character space.
 * This can help to better align the output of commands like `G29 O` Mesh Output.
 *
 * For clients that use a fixed-width font (like OctoPrint), leave this set to 1.0.
 * Otherwise, adjust according to your client and font.
 */
#define PROPORTIONAL_FONT_RATIO 1.0

/**
 * Spend 28 bytes of SRAM to optimize the GCode parser
 */
#define FASTER_GCODE_PARSER

/**
 * User-defined menu items that execute custom GCode
 */
//#define CUSTOM_USER_MENUS
#if ENABLED(CUSTOM_USER_MENUS)
  #define USER_SCRIPT_DONE "M117 User Script Done"
  #define USER_SCRIPT_AUDIBLE_FEEDBACK

  #define USER_DESC_1 "Home & UBL Info"
  #define USER_GCODE_1 "G28\nG29 W"

  #define USER_DESC_2 "Preheat for PLA"
  #define USER_GCODE_2 "M140 S" STRINGIFY(PREHEAT_1_TEMP_BED) "\nM104 S" STRINGIFY(PREHEAT_1_TEMP_HOTEND)

  #define USER_DESC_3 "Preheat for ABS"
  #define USER_GCODE_3 "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nM104 S" STRINGIFY(PREHEAT_2_TEMP_HOTEND)

  #define USER_DESC_4 "Heat Bed/Home/Level"
  #define USER_GCODE_4 "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nG28\nG29"

  #define USER_DESC_5 "Home & Info"
  #define USER_GCODE_5 "G28\nM503"
#endif

/**
 * Specify an action command to send to the host when the printer is killed.
 * Will be sent in the form '//action:ACTION_ON_KILL', e.g. '//action:poweroff'.
 * The host must be configured to handle the action command.
 */
//#define ACTION_ON_KILL "poweroff"

//===========================================================================
//===================== I2C Position Encoder Settings =======================
//===========================================================================

/**
 *  I2C position encoders for closed loop control.
 *  Developed by Chris Barr at Aus3D.
 *
 *  Wiki: http://wiki.aus3d.com.au/Magnetic_Encoder
 *  Github: https://github.com/Aus3D/MagneticEncoder
 *
 *  Supplier: http://aus3d.com.au/magnetic-encoder-module
```

```
 *  Alternative Supplier: http://reliabuild3d.com/
 *
 *  Reilabuild encoders have been modified to improve reliability.
 */

//#define I2C_POSITION_ENCODERS
#if ENABLED(I2C_POSITION_ENCODERS)

  #define I2CPE_ENCODER_CNT         1                     // The number of encoders installed; max of 5
                                                          // encoders supported currently.

  #define I2CPE_ENC_1_ADDR          I2CPE_PRESET_ADDR_X   // I2C address of the encoder. 30-200.
  #define I2CPE_ENC_1_AXIS          X_AXIS                // Axis the encoder module is installed on.
<X|Y|Z|E>_AXIS.
  #define I2CPE_ENC_1_TYPE          I2CPE_ENC_TYPE_LINEAR // Type of encoder:  I2CPE_ENC_TYPE_LINEAR -
or-
                                                          // I2CPE_ENC_TYPE_ROTARY.
  #define I2CPE_ENC_1_TICKS_UNIT    2048                  // 1024 for magnetic strips with 2mm poles;
2048 for
                                                          // 1mm poles. For linear encoders this is
ticks / mm,
                                                          // for rotary encoders this is ticks /
revolution.
  //#define I2CPE_ENC_1_TICKS_REV     (16 * 200)          // Only needed for rotary encoders; number of
stepper
                                                          // steps per full revolution (motor steps/rev
* microstepping)
  //#define I2CPE_ENC_1_INVERT                            // Invert the direction of axis travel.
  #define I2CPE_ENC_1_EC_METHOD     I2CPE_ECM_NONE        // Type of error error correction.
  #define I2CPE_ENC_1_EC_THRESH     0.10                  // Threshold size for error (in mm) above
which the
                                                          // printer will attempt to correct the error;
errors
                                                          // smaller than this are ignored to minimize
effects of
                                                          // measurement noise / latency (filter).

  #define I2CPE_ENC_2_ADDR          I2CPE_PRESET_ADDR_Y   // Same as above, but for encoder 2.
  #define I2CPE_ENC_2_AXIS          Y_AXIS
  #define I2CPE_ENC_2_TYPE          I2CPE_ENC_TYPE_LINEAR
  #define I2CPE_ENC_2_TICKS_UNIT    2048
  //#define I2CPE_ENC_2_TICKS_REV   (16 * 200)
  //#define I2CPE_ENC_2_INVERT
  #define I2CPE_ENC_2_EC_METHOD     I2CPE_ECM_NONE
  #define I2CPE_ENC_2_EC_THRESH     0.10

  #define I2CPE_ENC_3_ADDR          I2CPE_PRESET_ADDR_Z   // Encoder 3.  Add additional configuration
options
  #define I2CPE_ENC_3_AXIS          Z_AXIS                // as above, or use defaults below.

  #define I2CPE_ENC_4_ADDR          I2CPE_PRESET_ADDR_E   // Encoder 4.
  #define I2CPE_ENC_4_AXIS          E_AXIS

  #define I2CPE_ENC_5_ADDR          34                    // Encoder 5.
  #define I2CPE_ENC_5_AXIS          E_AXIS

  // Default settings for encoders which are enabled, but without settings configured above.
  #define I2CPE_DEF_TYPE            I2CPE_ENC_TYPE_LINEAR
  #define I2CPE_DEF_ENC_TICKS_UNIT  2048
  #define I2CPE_DEF_TICKS_REV       (16 * 200)
  #define I2CPE_DEF_EC_METHOD       I2CPE_ECM_NONE
  #define I2CPE_DEF_EC_THRESH       0.1

  //#define I2CPE_ERR_THRESH_ABORT  100.0                 // Threshold size for error (in mm) error on
any given
                                                          // axis after which the printer will abort.
Comment out to
                                                          // disable abort behaviour.

  #define I2CPE_TIME_TRUSTED        10000                 // After an encoder fault, there must be no
further fault
                                                          // for this amount of time (in ms) before the
encoder
                                                          // is trusted again.

  /**
   * Position is checked every time a new command is executed from the buffer but during long moves,
   * this setting determines the minimum update time between checks. A value of 100 works well with
   * error rolling average when attempting to correct only for skips and not for vibration.
   */
  #define I2CPE_MIN_UPD_TIME_MS     100                   // Minimum time in miliseconds between
```

```
encoder checks.

  // Use a rolling average to identify persistant errors that indicate skips, as opposed to vibration and
noise.
  #define I2CPE_ERR_ROLLING_AVERAGE

#endif // I2C_POSITION_ENCODERS

#endif // CONFIGURATION_ADV_H
```