```
/****************************************************************************\
*                                                                            *
* 1. CPU                                                                     *
*                                                                            *
\****************************************************************************/

/** \def CPU_TYPE
  CPU types a user should be able to choose from in configtool. All
  commented out.
*/
//#define CPU_TYPE atmega644
//#define CPU_TYPE atmega644p
//#define CPU_TYPE atmega1284
//#define CPU_TYPE atmega1284p

/** \def CPU
  CPU actually present on the board.
*/
#define CPU                     atmega644p

/** \def F_CPU_OPT
  CPU clock frequencies a user should be able to choose from in configtool.
  All commented out.
*/
//#define F_CPU_OPT 16000000UL
//#define F_CPU_OPT 20000000UL

/** \def F_CPU
  Actual CPU clock rate. #ifndef required for Arduino compatibility.
*/
#ifndef F_CPU
#define F_CPU                   20000000UL
#endif

/** \def MOTHERBOARD
  This is the motherboard, as opposed to the extruder. See extruder/ directory
  for GEN3 extruder firmware.
*/
#define MOTHERBOARD


/****************************************************************************\
*                                                                            *
* 2. PINOUTS                                                                 *
*                                                                            *
\****************************************************************************/

#define TX_ENABLE_PIN           DIO12
#define RX_ENABLE_PIN           DIO13
```

```
#define X_STEP_PIN             DIO19
#define X_DIR_PIN              DIO18
#define X_MIN_PIN              DIO0
//#define X_MAX_PIN             DIO6
//#define X_ENABLE_PIN           xxxx
//#define X_INVERT_DIR
//#define X_INVERT_MIN
//#define X_INVERT_MAX
//#define X_INVERT_ENABLE

#define Y_STEP_PIN             DIO21
#define Y_DIR_PIN              DIO20
//#define Y_MIN_PIN             DIO1
#define Y_MAX_PIN              DIO1
//#define Y_ENABLE_PIN           xxxx
//#define Y_INVERT_DIR
//#define Y_INVERT_MIN
//#define Y_INVERT_MAX
//#define Y_INVERT_ENABLE

#define Z_STEP_PIN             DIO23
#define Z_DIR_PIN              DIO22
#define Z_MIN_PIN              DIO2
//#define Z_MAX_PIN             DIO0
//#define Z_ENABLE_PIN           xxxx
#define Z_INVERT_DIR
//#define Z_INVERT_MIN
//#define Z_INVERT_MAX
//#define Z_INVERT_ENABLE

#define E_STEP_PIN             DIO28
#define E_DIR_PIN              DIO27
//#define E_ENABLE_PIN           xxxx
//#define E_INVERT_DIR
//#define E_INVERT_ENABLE

//#define PS_ON_PIN              DIO15
//#define PS_INVERT_ON
//#define PS_MOSFET_PIN          xxxx
#define STEPPER_ENABLE_PIN      DIO24
#define STEPPER_INVERT_ENABLE

/** \def DEBUG_LED_PIN

  Enable flashing of a LED during motor stepping.

  Disabled by default. Uncommenting this makes the binary a few bytes larger
  and adds a few cycles to the step timing interrrupt in timer.c. Also used
  for precision profiling (profiling works even without actually having such
```

```
  a LED in hardware), see
  http://reprap.org/wiki/Teacup_Firmware#Doing_precision_profiling
*/
//#define DEBUG_LED_PIN           DIO21

/** \def SD_CARD_SELECT_PIN

  Chip Select pin of the SD card.

  SD cards work over SPI and have a Chip Select or Slave Select (SS) pin.
  Choose this pin according to where on the board your SD card adapter is
  connected. Disabling this pin also disables SD card support and makes the
  firmware binary about 4.5 kB smaller.

  Connecting a device to SPI actually uses 4 signal lines, the other three
  pins are choosen by Teacup automatically.
*/
#define SD_CARD_SELECT_PIN      DIO25

/** \def MCP3008_SELECT_PIN

  Chip Select pin of the MCP3008 ADC.

  MCP3008/4 analog-digital converter works over SPI and has a Chip Select pin.
  Choose this pin according to where the MCP3008 is connected. Setting this
  pin is required only if at least one temperature sensor of type MCP3008 is
  configured. Else it's ignored.
*/
//#define MCP3008_SELECT_PIN      xxxx


/***************************************************************************\
*                                                                           *
* 3. TEMPERATURE SENSORS                                                    *
*                                                                           *
\***************************************************************************/

#ifndef DEFINE_TEMP_SENSOR
  #define DEFINE_TEMP_SENSOR(...)
#endif

/** \def TEMP_MAX6675 TEMP_THERMISTOR TEMP_AD595 TEMP_PT100 TEMP_INTERCOM
    \def TEMP_MCP3008

  Which temperature sensor types are you using? Leave all used ones
  uncommented, comment out all others to save binary size and enhance
  performance.
*/
//#define TEMP_MAX6675
#define TEMP_THERMISTOR
```

```
#define TEMP_AD595
//#define TEMP_PT100
//#define TEMP_INTERCOM
//#define TEMP_MCP3008

/** \def TEMP_SENSOR_PIN
  Temperature sensor pins a user should be able to choose from in configtool.
  All commented out.
*/
//#define TEMP_SENSOR_PIN AIO1
//#define TEMP_SENSOR_PIN AIO2

/** \def DEFINE_TEMP_SENSOR
  Define your temperature sensors here. One line for each sensor, only
  limited by the number of available ATmega pins.

  Name must match the name of the corresponding heater. If a heater "extruder"
  exists, a temperature sensor of that name has to exist as well. Same for
  heater "bed". There can be one sensor without corresponding heater, name it
  "noheater".

  Types are same as TEMP_ list above - TT_MAX6675, TT_THERMISTOR, TT_AD595,
  TT_PT100, TT_INTERCOM, TT_MCP3008. See list in temp.c.

  The "additional" field is used for TT_THERMISTOR and TT_MCP3008 only. It
  defines the name of the table(s) in thermistortable.h to use. This name is
  arbitrary, often used names include THERMISTOR_EXTRUDER and THERMISTOR_BED.
  Also, several sensors can share the same table, which saves binary size.

  For a GEN3 set temp_type to TT_INTERCOM and temp_pin to AIO0. The pin
  won't be used in this case.
*/
//DEFINE_TEMP_SENSORS_START
//              name      type          pin    additional
DEFINE_TEMP_SENSOR(extruder, TT_THERMISTOR, AIO1,  THERMISTOR_EXTRUDER)
DEFINE_TEMP_SENSOR(bed,      TT_AD595,      AIO2,  0)

// Beta algorithm      r0      beta  r2    vadc
// Steinhart-Hart      rp      t0    r0      t1    r1      t2    r2
//TEMP_TABLE EXTRUDER (200000, 4338, 10000,5.0)
//DEFINE_TEMP_SENSORS_END


/*************************************************************************\
*                                                                         *
* 4. HEATERS                                                              *
*                                                                         *
\*************************************************************************/

/** \def FORCE_SOFTWARE_PWM
```

```
  Force software pwm when pwm is sets to 1.

  Normally any pwm value >= 1 will set the pin to hardware pwm, if available.
  When FORCE_SOFTWARE_PWM is defined, pwm = 1 is always set to software pwm.
*/
// #define FORCE_SOFTWARE_PWM

/** \def HEATER_PIN
  Heater pins a user should be able to choose from in configtool. All
  commented out.
*/
//#define HEATER_PIN DIO3
//#define HEATER_PIN DIO4
//#define HEATER_PIN DIO15

/** \def DEFINE_HEATER
  Define your heaters and devices here.

  To attach a heater to a temp sensor above, simply use exactly the same
  name - copy+paste is your friend. Some common names are 'extruder',
  'bed', 'fan', 'motor', ... names with special meaning can be found
  in gcode_process.c. Currently, these are:
    HEATER_extruder   (M104)
    HEATER_bed        (M140)
    HEATER_fan        (M106)

  Devices don't neccessarily have a temperature sensor, e.g. fans or
  milling spindles. Operate such devices by setting their power (M106),
  instead of setting their temperature (M104).

  Also note, the index of a heater (M106 P#) can differ from the index of
  its attached temperature sensor (M104 P#) in case sensor-less devices
  are defined or the order of the definitions differs. The first defined
  device has the index 0 (zero).

  Set 'invert' to 0 for normal heaters. Setting it to 1 inverts the pin signal
  for this pin, e.g. for a MOSFET with a driver.

  Set 'pwm' to ...
    1  for using hardware PWM on a PWM-able pin and software PWM on other pins.
    0  for using on/off on a PWM-able pin, too.

  Using PWM usually gives smoother temperature control but can conflict
  with slow switches, like solid state relays. PWM frequency can be
  influenced globally with FAST_PWM, see below.
*/

//DEFINE_HEATERS_START
//          name      pin      invert  pwm     max_pwm
DEFINE_HEATER(extruder, DIO4,    0,      1,     100)
```

```
DEFINE_HEATER(bed,     DIO3,   0,     0,     100)
DEFINE_HEATER(fan,     DIO15,  0,     1,     100)

#define HEATER_EXTRUDER HEATER_extruder
#define HEATER_BED HEATER_bed
#define HEATER_FAN HEATER_fan
//DEFINE_HEATERS_END


/**************************************************************************\
*                                                                          *
* 5. COMMUNICATION OPTIONS                                                 *
*                                                                          *
\**************************************************************************/

/** \def BAUD
  Baud rate for the serial RS232 protocol connection to the host. Usually
  115200, other common values are 19200, 38400 or 57600. Ignored when USB_SERIAL
  is defined.
*/
#define BAUD                    57600

/** \def XONXOFF
  Xon/Xoff flow control.

  Redundant when using RepRap Host for sending G-code, but mandatory when
  sending G-code files with a plain terminal emulator, like GtkTerm (Linux),
  CoolTerm (Mac) or HyperTerminal (Windows).
*/
//#define XONXOFF

/** \def USB_SERIAL
  Define this for using USB instead of the serial RS232 protocol. Works on
  USB-equipped ATmegas, like the ATmega32U4, only.
*/
//#define USB_SERIAL


/**************************************************************************\
*                                                                          *
* 6. DISPLAY SUPPORT                                                       *
*                                                                          *
\**************************************************************************/

/** \def DISPLAY_BUS_4BIT DISPLAY_BUS_8BIT DISPLAY_BUS_I2C DISPLAY_BUS_SPI

  The bus used to connect the display to the controller. This is a property
  of the display. With most displays there can be only one correct choice.

  Comment in the one in use, comment out all others. If there is no display,
```

```
  comment out all of them to remove display code for better performance.
*/
#define DISPLAY_BUS_4BIT
//#define DISPLAY_BUS_8BIT
//#define DISPLAY_BUS_I2C
//#define DISPLAY_BUS_SPI

/** \def DISPLAY_RS_PIN DISPLAY_RW_PIN DISPLAY_E_PIN
    \def DISPLAY_D4_PIN DISPLAY_D5_PIN DISPLAY_D6_PIN DISPLAY_D7_PIN

  Pins necessary for the 4-bit parallel display bus. Taken into account with
  DISPLAY_BUS_4BIT defined, only.
*/
#define DISPLAY_RS_PIN          PC1
#define DISPLAY_RW_PIN          PC0
#define DISPLAY_E_PIN           PD2
#define DISPLAY_D4_PIN          PD3
#define DISPLAY_D5_PIN          PD4
#define DISPLAY_D6_PIN          PD5
#define DISPLAY_D7_PIN          PD6

/** \def DISPLAY_TYPE_SSD1306 DISPLAY_TYPE_HD44780

  The type of display in use. There can be only one choice. Taken into account
  only if one of DISPLAY_BUS_xxx is defined.

  Comment in the display in use, comment out all others. If there is no
  display, comment out all of DISPLAY_BUS_xxx.
*/
//#define DISPLAY_TYPE_SSD1306
#define DISPLAY_TYPE_HD44780
```