K5BCQ K3NG Keyer v0.6

Josh (W0ODJ), Ken (KM4NFQ)

July 15, 2019

Contents

1 Introduction

This keyer is a design of the proposed schematic provided by Anthony Good (K3NG) on his website, Radioartisan. As taken from his website:

> "This is an open source Arduino based CW (Morse Code) keyer with a lot of features and flexibility, rivaling commercial keyers which often cost significantly more. The code can be used with a full blown Arduino board or an AVR microcontroller chip can be programmed and used directly in a circuit. This keyer is suitable as a standalone keyer or for use permanently installed inside a rig, especially homebrew QRP rigs. It's open source code so you can fully customize it to fit your needs and also perhaps learn from it or find coding ideas for other projects."

This board is an attempt to provide an outline for utilizing many of the options available in Goody's amazing open-source arduino-based morse code keyer. Mechanisms are available for multiple size LCDs, I2C displays, 2 transmitter outputs, a Goertzel audio decoding input, rotary encoder for speed control, seven memories, hookups for both 3x4 and 4x4 number pads, amongst others. It is designed to be built for only the modules you need - you do not need to populate those you are not going to use!

This documentation is specific for the K5BCQ board of the K3NG keyer. However, in most locations you can change the hardware specific reference (e.g. *"keyer_pin_settings_k5bcq.h"* to the generic reference

"keyer_pin_settings.h") relatively easily. That means the comments regarding pinouts, enabling features, etc - should all be mostly universal (though specific pins, etc. will likely change).

The PCB images of part locations have been painstakingly created by Ken, KM4NFQ. He has graciously allowed them to be included in this documentation.

2 Bill of Materials

Reference	Value	Quantity	Digikey #	Notes
R1, 2, 3, 4, 5, 6, 7	1k	7	CF18JT1K00CT-ND	1/8w
R12, 18, 27, 28	1k	4	CF14JT1K00CT-ND	1/4w
R16, 17, 21, 22	10k	5	CF14JT10K0CT-ND	1/4w
R9	10k	1	CF18JT10K0CT-ND	1/8w
R15	220	1	CF18JT220RCT-ND	1/8w
R13, 19, 20, 26	220	4	CF14JT220RCT-ND	1/4w
R14, 23, 36	100	3	CF14JT100RCT-ND	1/4w
R24, 25	4.7k	2	CF14JT4K70CT-ND	1/4w
R30, 31, 33	3.3k	3	CF18JT3K30CT-ND	1/8w
R32	560	1	CF18JT560RCT-ND	1/8w
R34	10	1	CF18JT10R0CT-ND	1/8w
R35	150k	1	CF18JT150KCT-ND	1/8w
R37	100k	1	CF18JT100KCT-ND	1/8w
R8, 11, 29	20k	3	3306F-203-ND	6mm top-adjust trimpot
R10	500	1	3306F-501-ND	6mm top-adjust trimpot
C1, 2, 3, 5, 7, 8	10nF	6	490-8634-ND	
C6, 9, 10, 11a, 11b, 12, 13	100nF	7	445-173396-1-ND	
C4	100uF	1	493-6095-1-ND	Electrolytic
C14, 15, 16, 17	10uF	4	P19580CT-ND	Electrolytic
Q1, 3, 4, 6, 8	2N7000	5	2N7000FS-ND	TO-92
Q2, 5, 7	2N3904	3	2N3904FS-ND	TO-92
LED1, 2, 3, 4		4		LED of your choice of colors
SW1		1	987-1398-ND	Rotary encoder with switch
SW2, 3, 4, 5, 6, 7, 8		7	450-1642-ND	6mm x 6mm NO tactile switch
J1, 2, 3, 4, 5		5	SC1461-ND	PJ-325M, These should work
J6		1	CP-002A-ND	DC plug
X1		1	CP-2260-ND	MiniDin 6 - MD-60SM
SP1		1		Off-board speaker
U1	Display	1		1602, 1604, 2004 Serial LCD
U2	Mega2560	1		Mini variant, here
U3	Keypad	1		3x3 or 3x4 keypad
P1, 2, 4A, 4P, ASR-JP				2.54mm Male headers

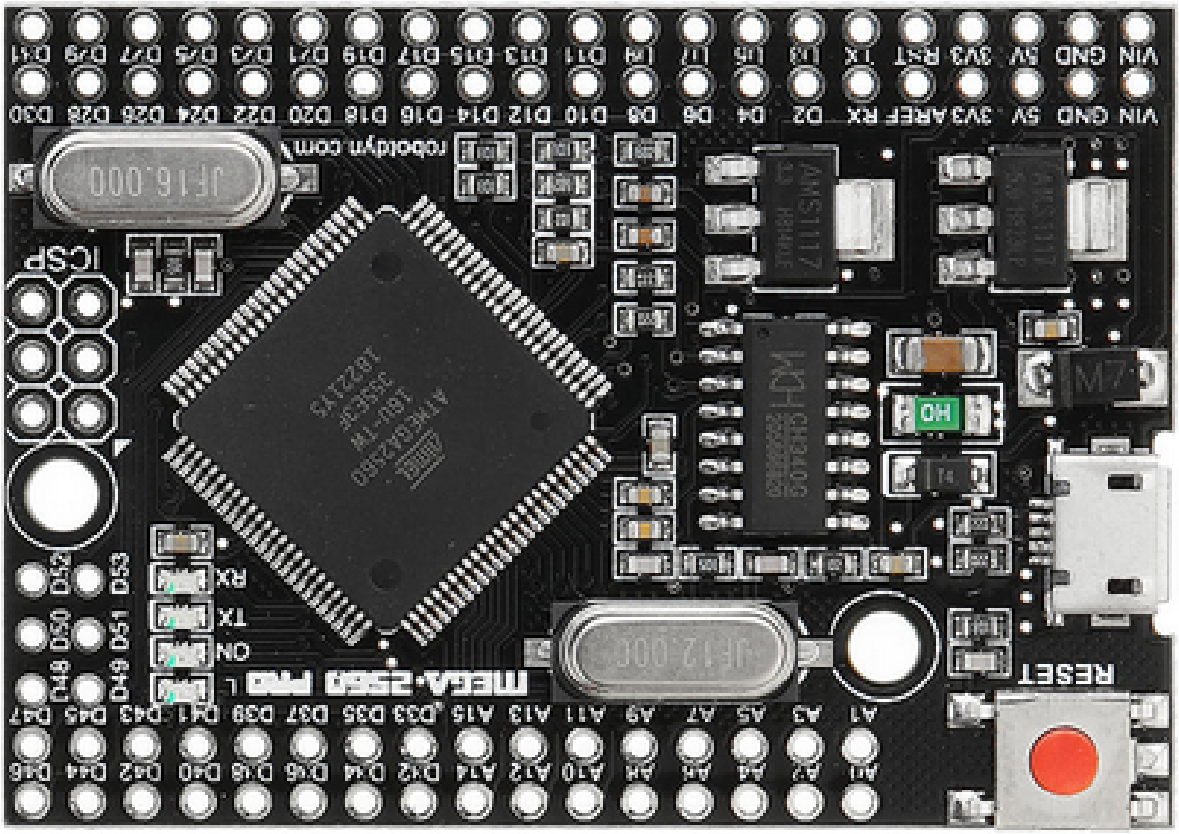
***** Part footprints should be correct - BUT I HAVE NOT CONFIRMED!!! Particularly capacitors lead spacing!***

2.1 Which Arduino?

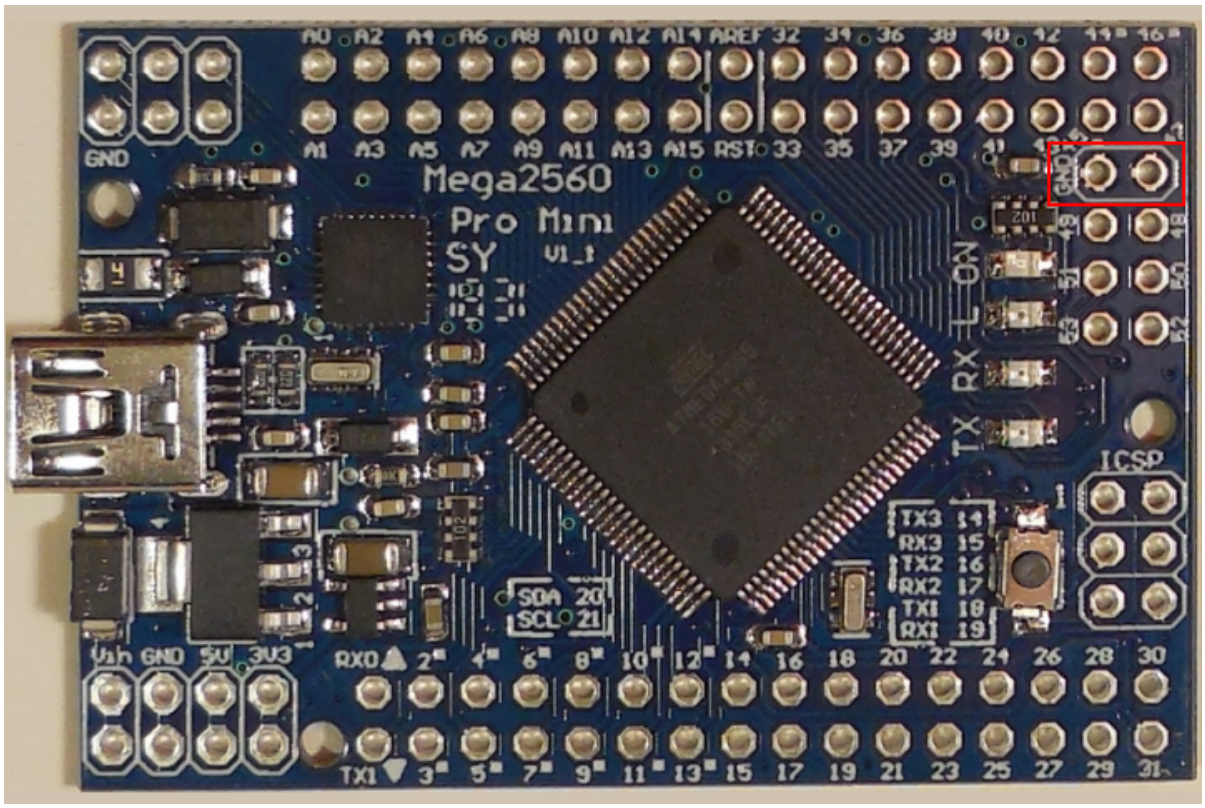
Of significant importance in the fact that while Arduino(s) is/are "standard," there is great leeway and interpretation in the particular details regarding "standard." When building the K5BCQ PCB of the K3NG keyer, you must make sure you are purchasing one of the boards with the correct footprint for the

PCB. While many producer's versions (may) work, there is no guarantee. Just be sure to obtain one with the correct voltage (5v), USB (if desired), and footprint. Linked above is a known good brand (RobotDyn) for the designed PCB.

2.1.1 The correct footprint looks like:



2.1.2 The incorrect footprint looks like:



The wrong pinout will *still* work, *technically* - but would require manually running jumper wires between the boards, instead of using headers for installation.

3 Build Instructions

Building is relatively straightforward. Instructions to build all available options is included in the documentation. If there is a module you do not wish to include, just skip that step. If there is an issue skipping the step, it will be noted in the documentation for that step.

If you are going to build all of the modules - ignore the following steps. Just build it normally - lowest parts first. My preferred order is resistors, capacitors, transistors, headers, buttons and other hardware.

3.1 Arduino

☐ Arduino

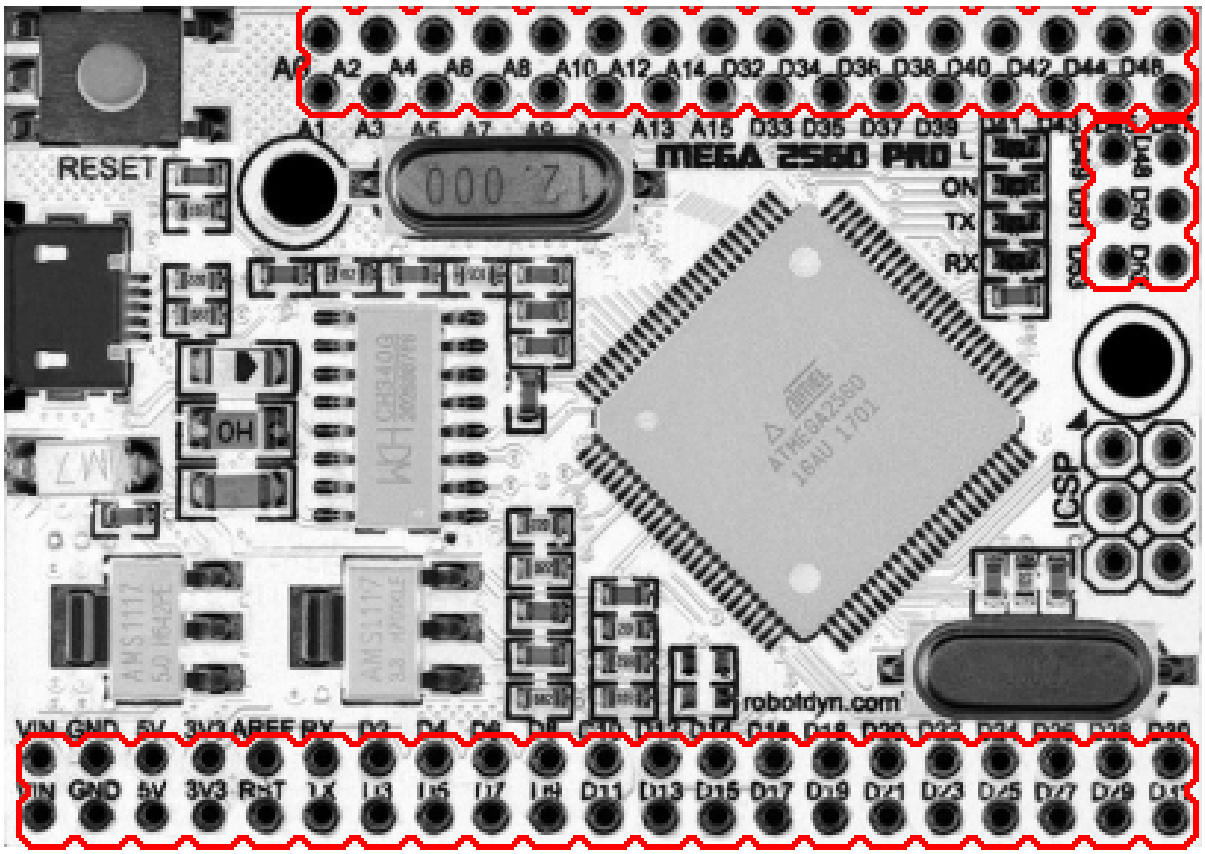
- ☐ Arduino Mega 2560 Pro Mini
- ☐ 2.54mm Male Headers (1 - 2x21, 1 - 2x16, 1 - 2x3)
- ☐ 2.54mm Female Headers (1 - 2x21, 1 - 2x16, 1 - 2x3)

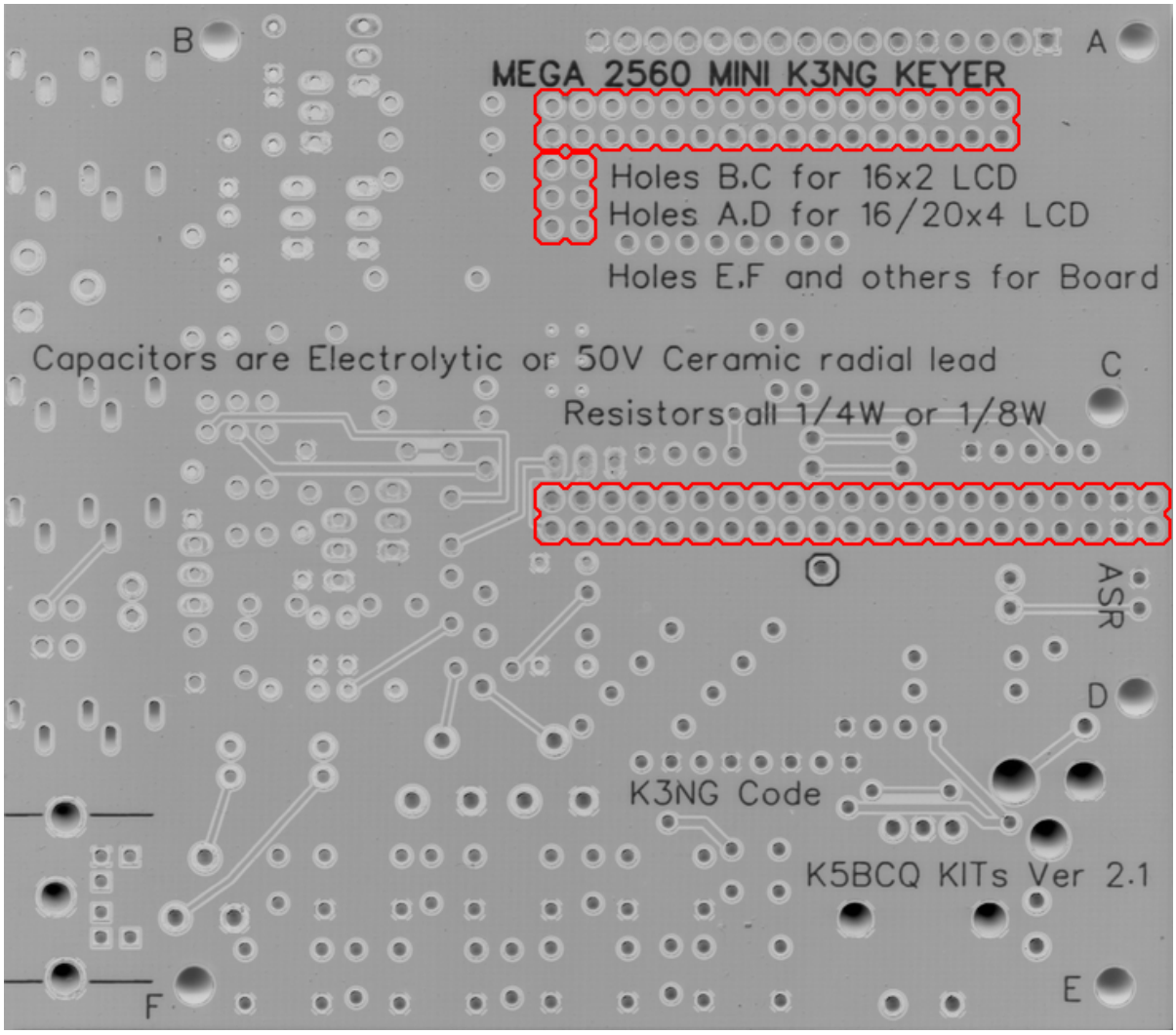
The Arduino is installed on the underside of the board. If you wish, you could directly solder the board to the main PCB using just the male headers - most people prefer the ability to remove the Arduino, however. It may be easier to install the Arduino and headers AFTER all of the other parts have been installed. Builder's discretion.

The easiest way to properly, neatly solder the board and headers, is to not solder any of it to begin with. After cutting/breaking/obtaining the right width and length of headers, start by placing the female headers on the underside of the board. Once they are in place, NOT SOLDERED, gently place the long portion of the male headers into them. Not all the way down - just into them. Once these are in place, then carefully align the male headers into the top of the Mega 2560. Once they are all in place, squeeze together. You now will have all of the pins in the correct placement, and they should all be almost perpendicular to the PCBs. Solder the pins at each end, adjust to make sure they are straight and flush to the PCBs, and then solder the remaining in-between pins.

Once all of the pins are soldered, remove the Arduino and set aside. You don't want to accidentally short it out.

Builder's Note: You can use single row headers (male and female) instead. Just be sure to carefully cut and align them, as well as ensure proper fit. Dual-row headers are preferred for easier mating / removal, but many individuals just use the single row headers side-by-side.


RESET
MEGA 2560 PRO
ON
TX
RX
ICSP
roboticdyn.com
VIN GND 5V 3V3 RST TX


B
A
MEGA 2560 MINI K3NG KEYER
Holes B.C for 16x2 LCD
Holes A.D for 16/20x4 LCD
Holes E.F and others for Board
Capacitors are Electrolytic or 50V Ceramic radial lead
C
Resistors all 1/4W or 1/8W
ASR
D
K3NG Code
K5BCQ KITs Ver 2.1
F
E

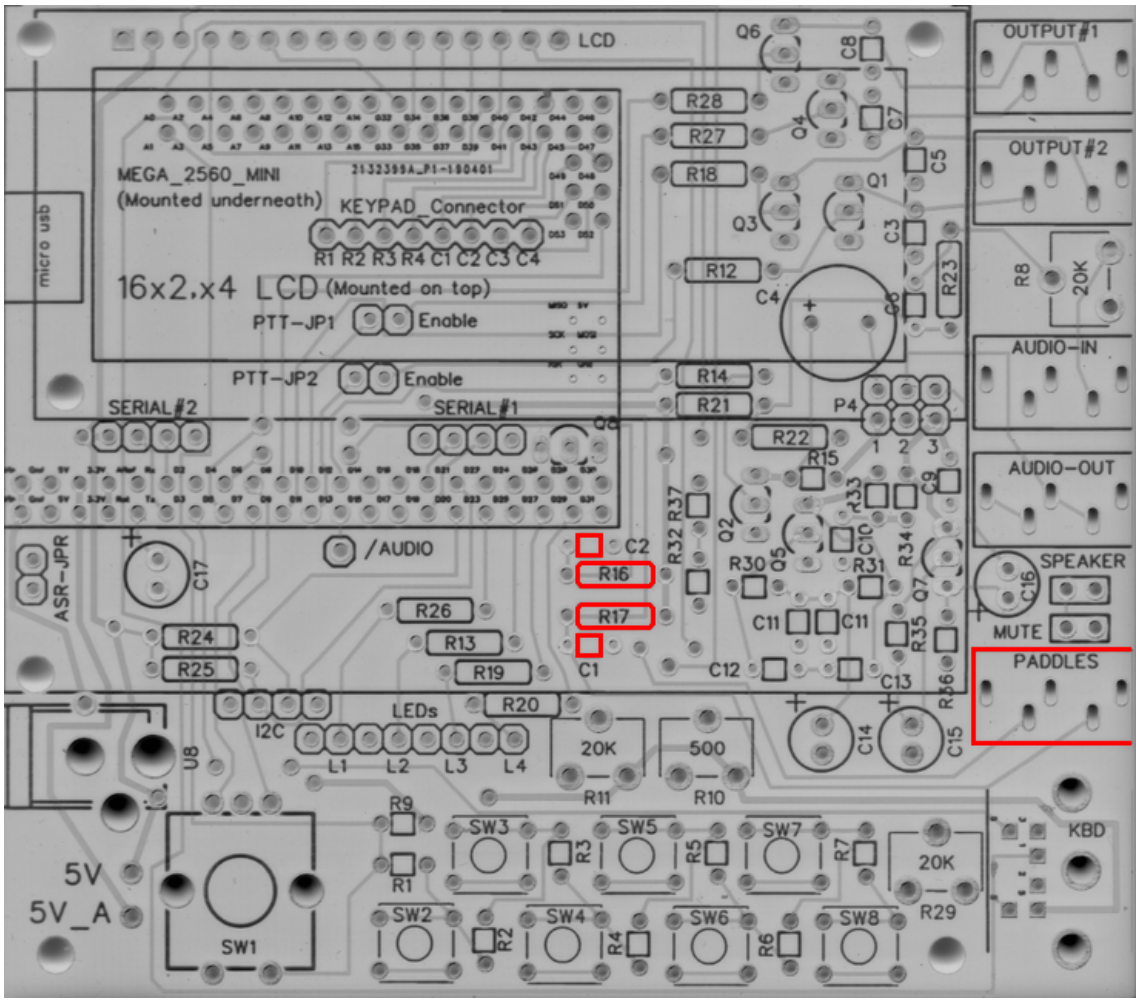
3.2 Inputs

☐ Paddle Input

- ☐ R16, 17 - 10kΩ - (Brown-Black-Orange)
- ☐ C1, 2 - 10nF - (103)
- ☐ JX - 3.5mm TRS audio jack

Input for dual lever paddles. Designed for the relatively standard 3.5mm TRS (Stereo) jack. Default wiring is Tip = Dit, Ring = Dah, Sleeve = Ground. Paddle wiring can be artificaly reversed in software using `"N"` in command mode (and `"\N"` in command line interface). Using the TRS jack is superior to Mono - it allows both paddle input, as well as Straight Key input. If you are only going to use Straight Key mode, enable *"#define FEATURE_STRAIGHT_KEY"* in *"keyer_features_and_options_k5bcq.h"*. Paddle pins are set using *"#define paddle_left"* and *"#define paddle_right"* in *"keyer_pin_settings_k5bcq.h"*.

It is preferred however, to ignore the Straight key functionality (leaving the module disabled), as having it enabled can cause timing / keying issues when using paddles. Without the module enabled, if you boot the keyer with a properly wired straight key (or a paddle with one key grounded), the keyer will function in straight key mode. All the benefit, none of the problems! (unless you *really* need to switch between paddles and straight key that quickly)

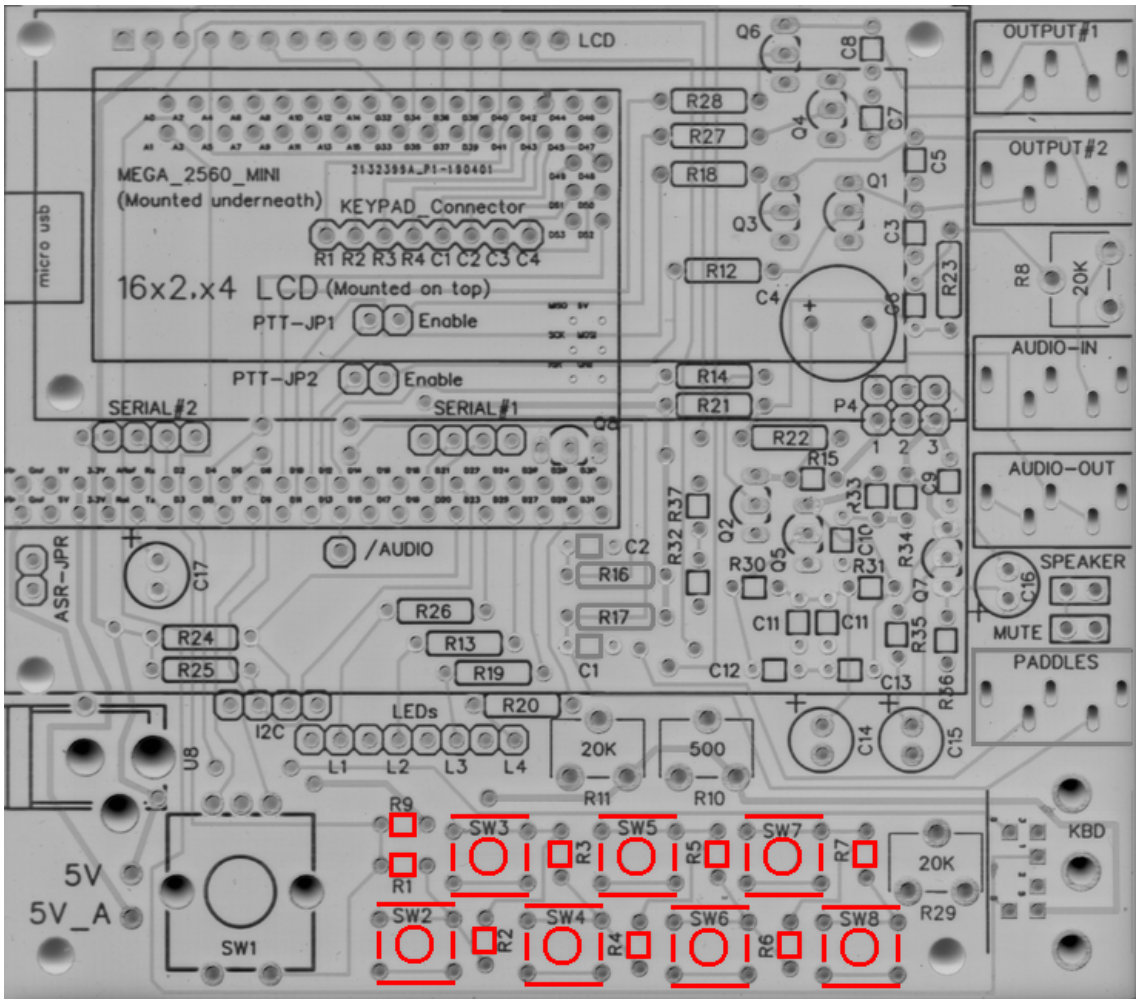


☐ Memory Buttons

- ☐ R1, 2, 3, 4, 5, 6, 7 - 1kΩ - (Brown-Black-Red)
- ☐ R9 - 10kΩ - (Brown-Black-Orange)
- ☐ SW2, 3, 4, 5, 6, 7, 8 - 6mm x 6mm NO Tactile push buttons

These buttons allow up to seven memories to be utilized. If you choose not to install them all, you need to edit *"FEATURE_COMMAND_BUTTONS"* in *"keyer_settings_k5bcq.h"* to accurately represent the number of buttons and the voltage divider used (r1 = R9, r2 = R1-7)

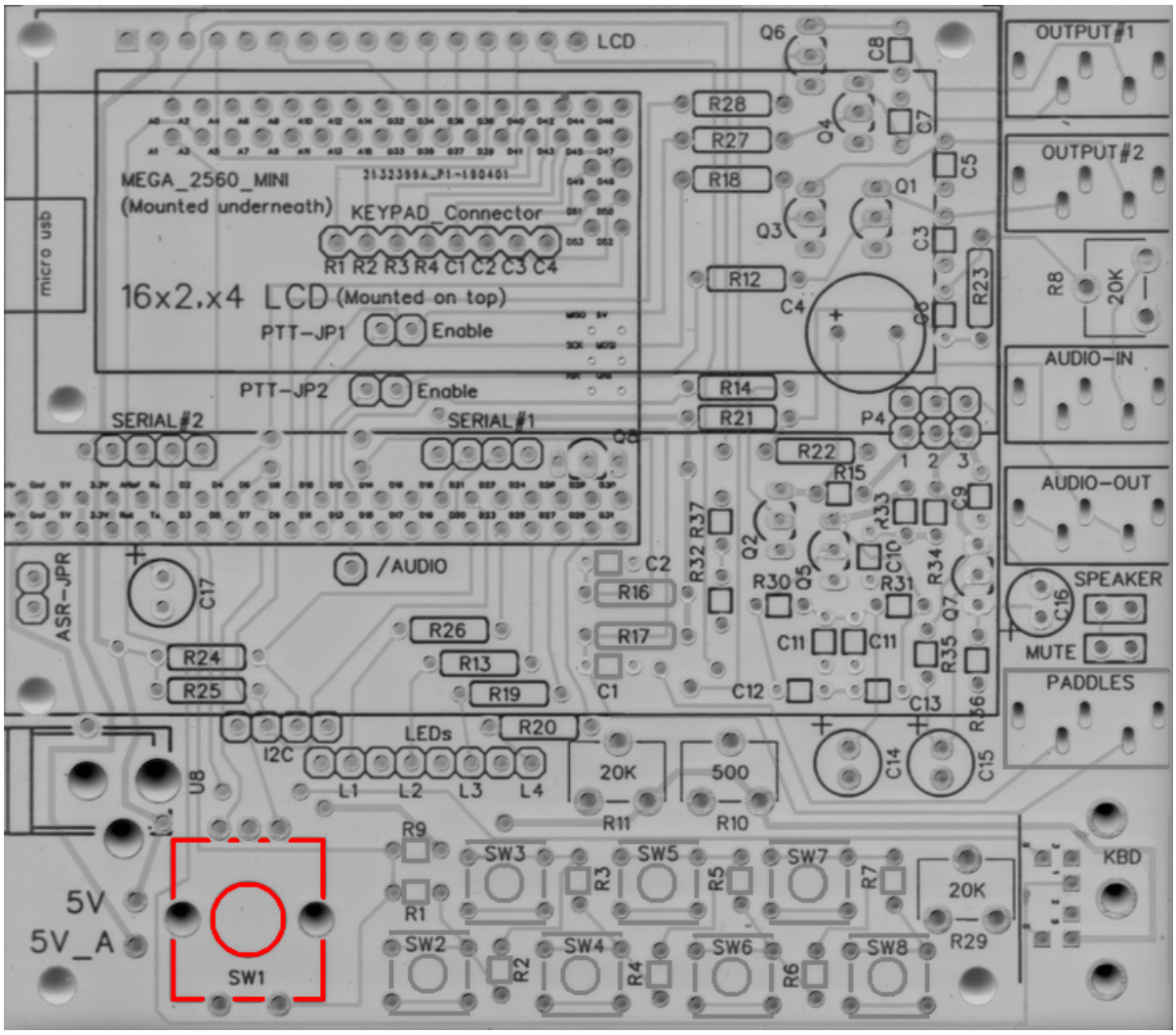
The height of the tactile button varies wildly based on your intended enclosure. You may wish to off-board them with wires or headers depending on your design / intention. For reference, the bottom two mounting holes of each button are ground.



☐ Rotary Encoder

☐ SW1 - Rotary encoder with integrated push button

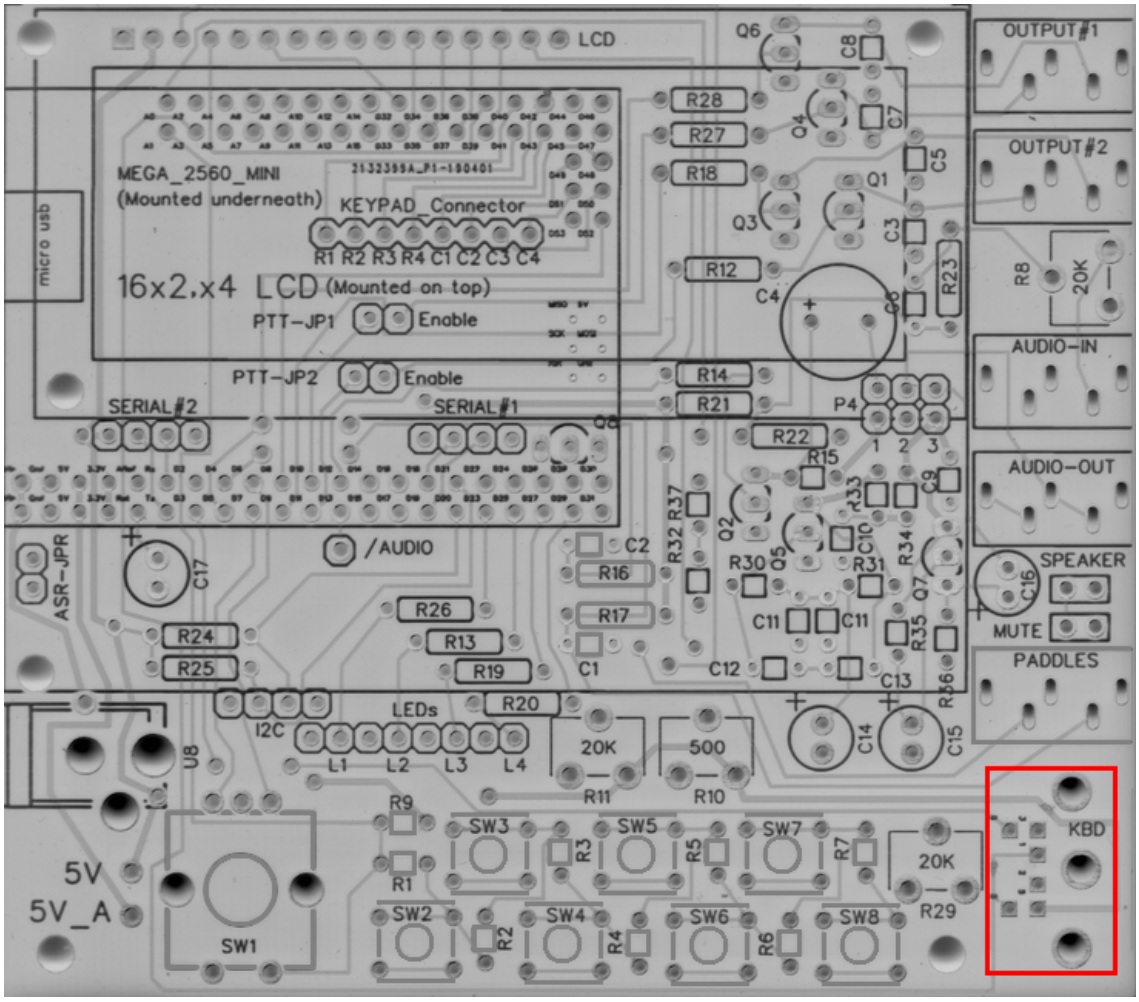
This encoder is used to adjust speed, and activate command mode (Button "1"). Depending on enclosure, you may wish to off-board this depending on mounting. Left to right, the pinouts are: Counter-Clockwise, Ground, Clockwise. These are configured using *"FEATURE_ROTARY_ENCODER"* in *"keyer_pin_settings_k5bcq.h"*. It is enabled by *"#define FEATURE_ROTARY_ENCODER"* in *"keyer_features_and_options_k5bcq.h"*.



☐ Keyboard Input

☐ X1 - MINI DIN 6

Keyboard input is enabled using a standard PS/2 keyboard MINI DIN 6 plug. No additional parts are required. A PS/2 keyboard is enabled using *"#define FEATURE_PS2_KEYBOARD"* in *"keyer_features_and_options_k5bcq.h"*. The library used for the keyboard is *"K3NG_PS2KEYBOARD"* which is included in the provided libraries. And yes, the library is uncategorized and will trigger a "warning" on compilation - you can safely ignore it.

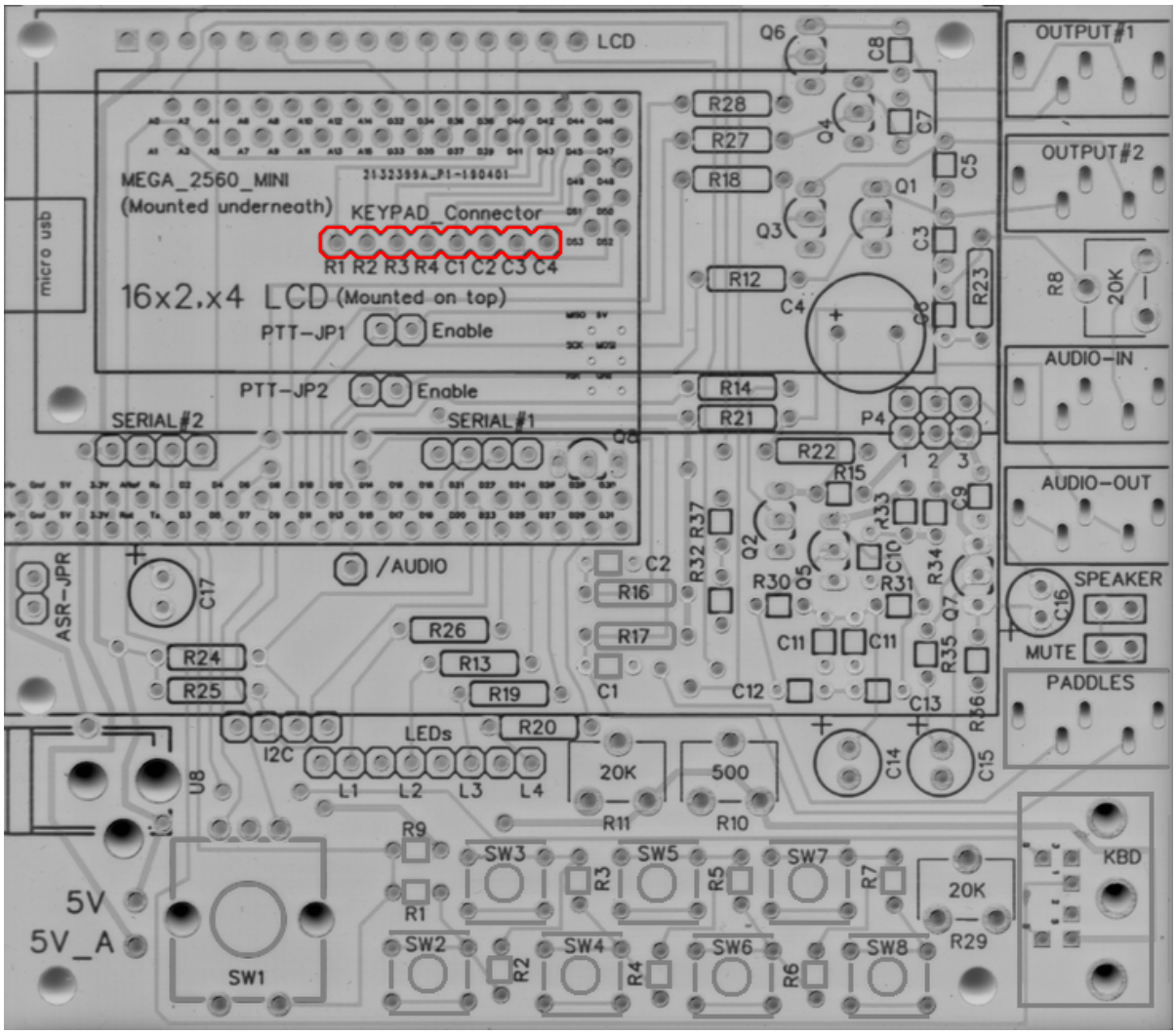


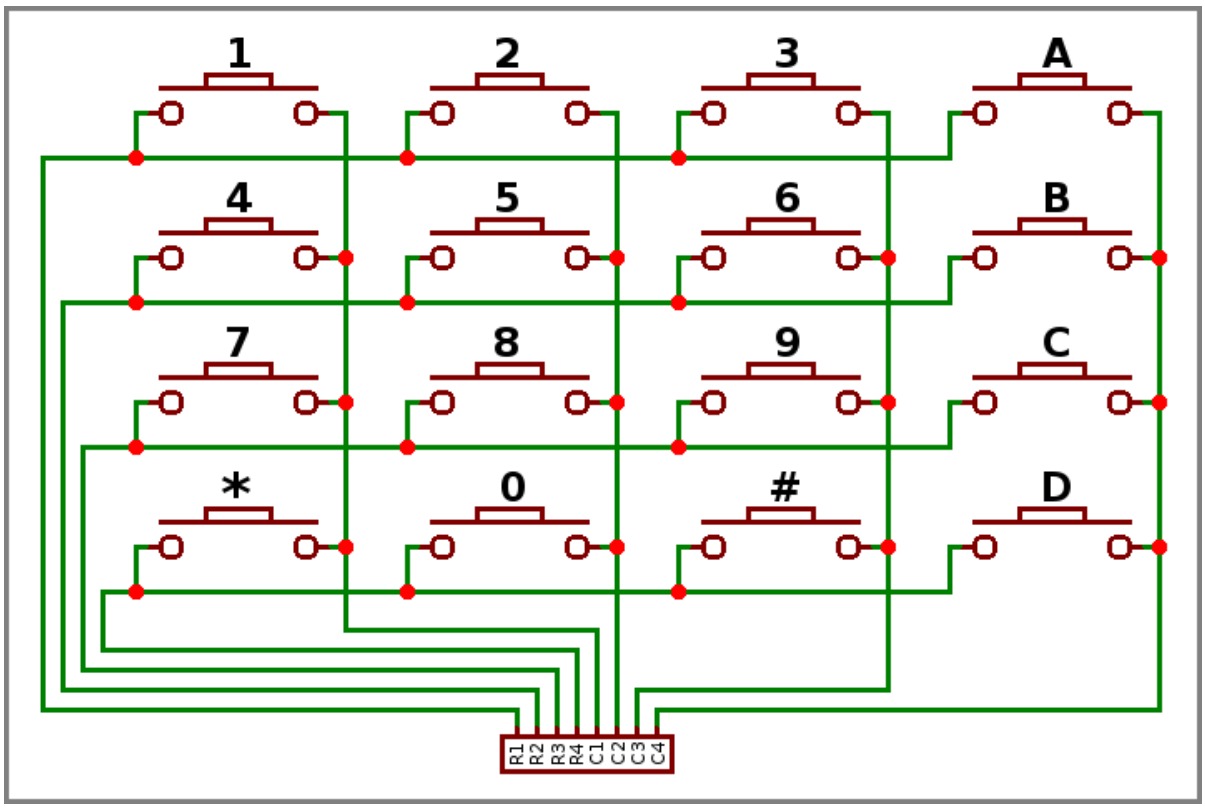
☐ Keypad

☐ U3 - 3x4 or 4x4 keypad

No additional parts are needed, other than the keypad. It is likely that you will off-board the connection to the keypad, and thus it is likely that you will prefer to use a 1x8 2.54mm male header for connections, though direct wiring can be done if you so choose.

To enable the use of the keypad, you **MUST** enable the appropriate module *"#define FEATURE_4x4KEYPAD"* or *"#define FEATURE_3x4_KEYPAD"* in *"keyer_features_and_options_k5bcq.h"*. The pinouts should be correct (matching the PCB markings), but no confirmation has been made at this time. The pins are set under *"FEATURE_4x4_KEYPAD"* and *"FEATURE_3x4_KEYPAD"* in *"keyer_pin_settings_k5bcq.h"*.



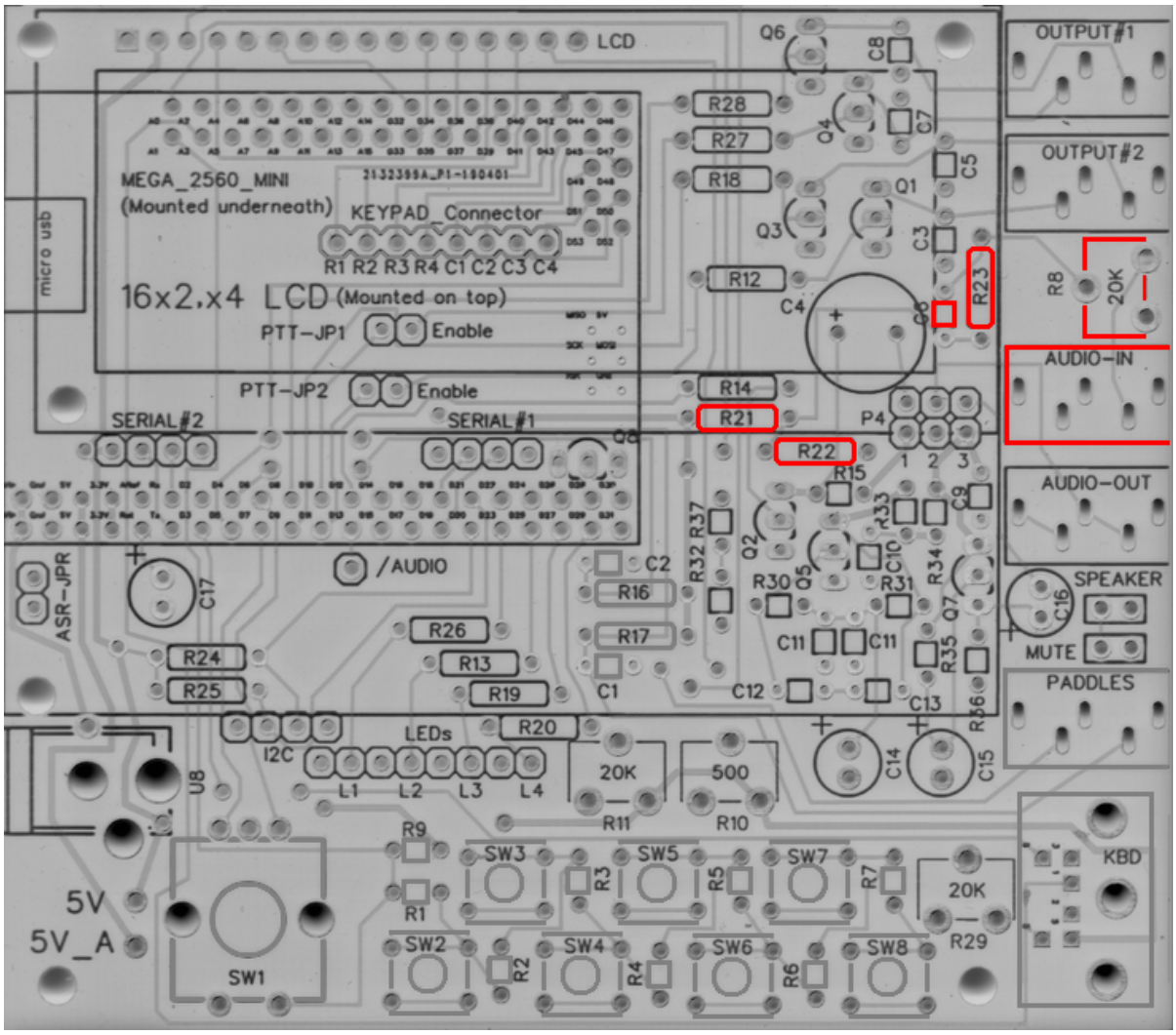

1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
R1
R2
R3
R4
C1
C2
C3
C4

☐ Goertzel Audio Decoder

- ☐ R8 - 20kΩ Trimpot - (203)
- ☐ R21, 22 - 10kΩ - (Brown-Black-Orange)
- ☐ R23 - 100Ω - (Brown-Black-Brown)
- ☐ C6 - 100nF - (104)
- ☐ J3 - 3.5mm TRS Jack

Audio input that is decoded using Goertzel algorithm. R8 is used to help set the audio voltage, and R11 and 22 bias the input voltage +/- 2.5volts.

Information regarding the Goertzel circuit and code used in the K3NG keyer can be found here. The variables for the Goertzel decoding must be set at compile time, and are NOT part of the sketch. If you wish to edit them, do so in *:goertzel.h:* in the K3NG libraries. The values for tweaking are *"GOERTZ_SAMPLES"* and *"GOERTZ_TARGET_FREQ"*. Editing these values is a trade off between precision and processing power. Please consult with the library for more details.

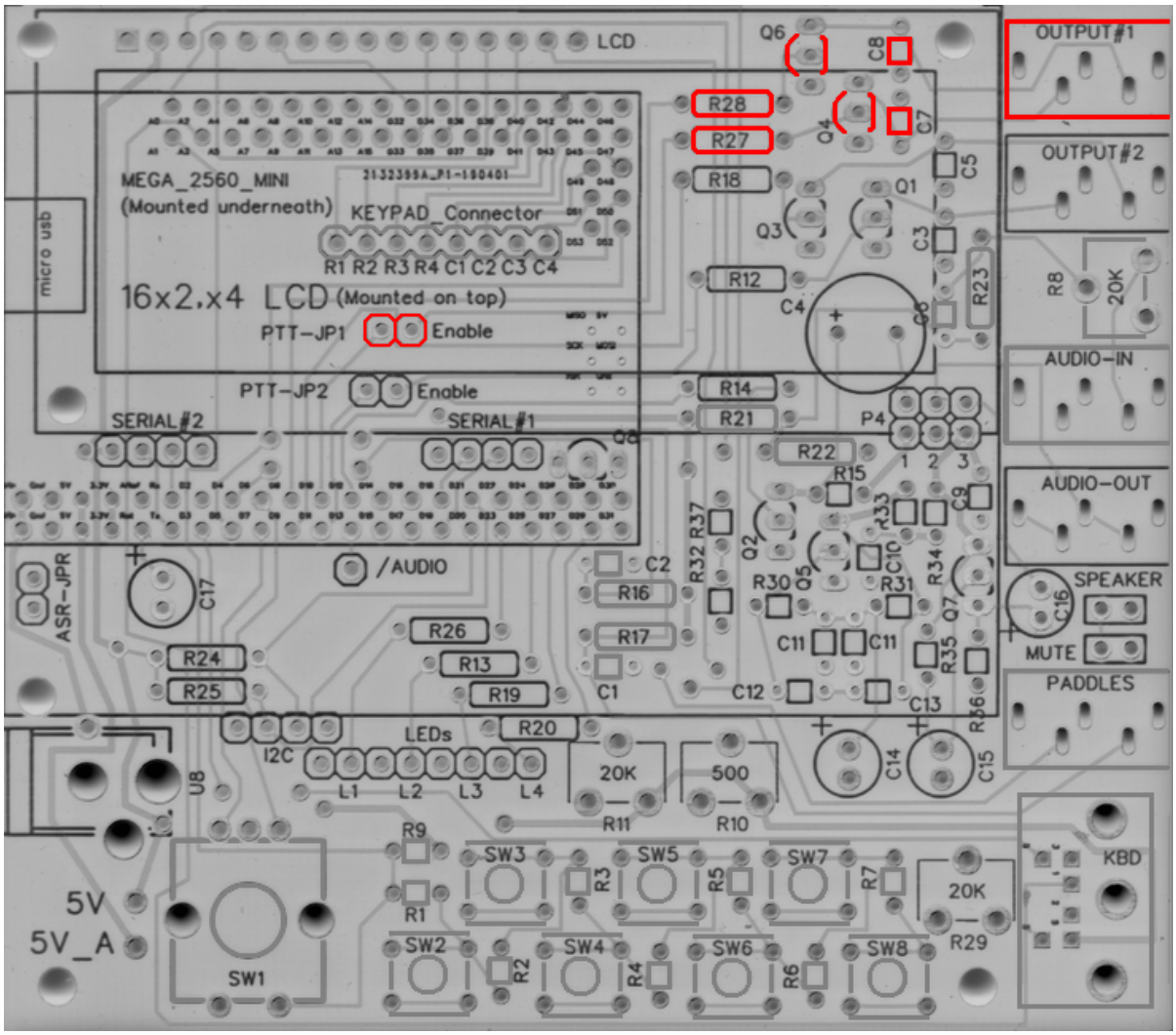


3.3 Outputs

☐ Keyer Output 1

- ☐ R27, 28 - 1kΩ - (Brown-Black-Red)
- ☐ C7, 8 - 10nF (103)
- ☐ Q4, Q6 - 2N7000 -
- ☐ JP1 - 1x2 2.54mm header with jumper
- ☐ JX - 3.5mm TRS audio jack

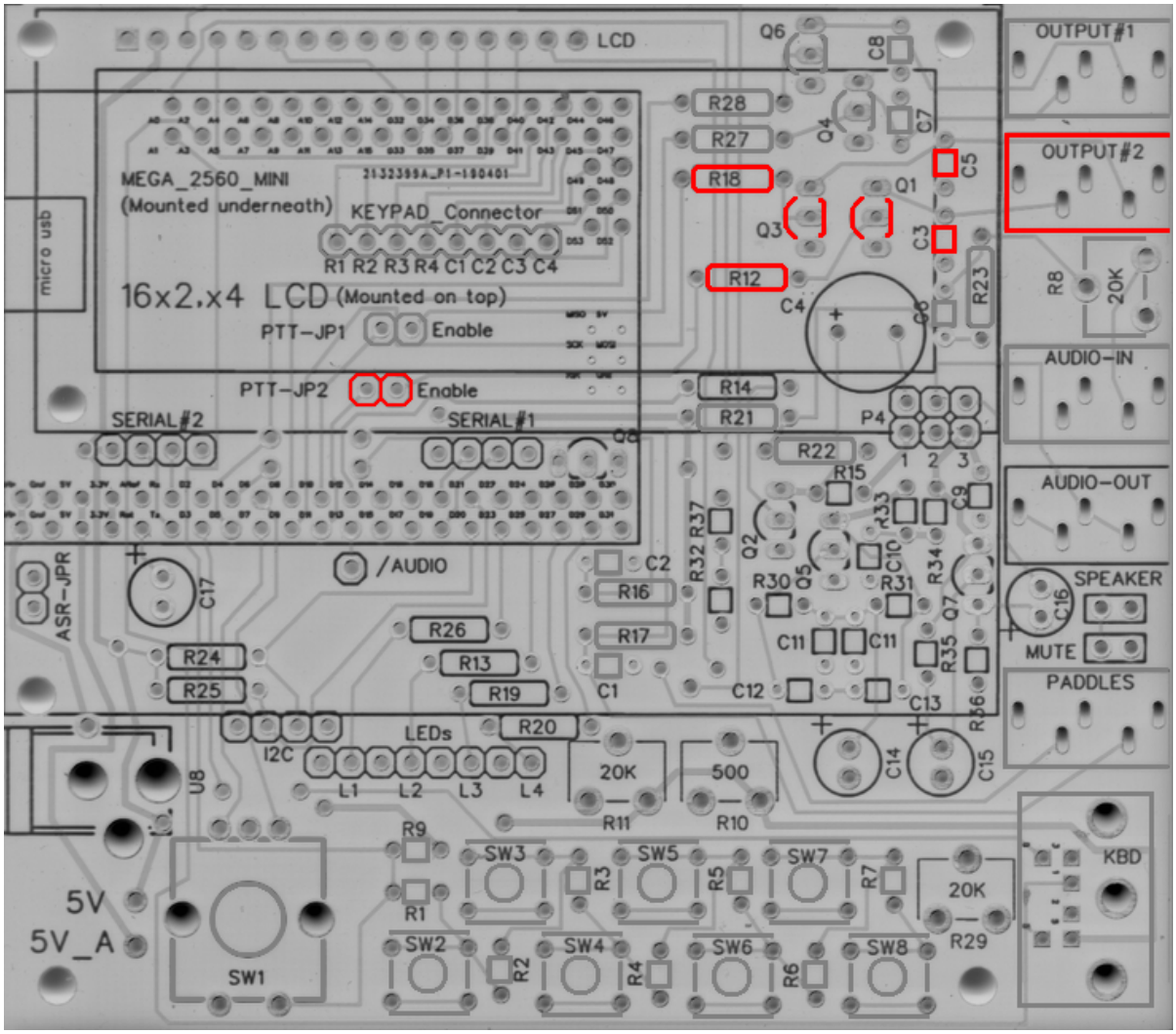
Keyer output with PTT. If you do not wish to be able to enable / disable PTT, you can permanently jumper JP1 with bus wire, a scrap resistor lead, or similar. Transmit key lines, and PTT line are configured by *"#define tx_key_line_1"* and *"#define ptt_tx_1"* respectively, in *"keyer_pin_settings_k5bcq.h"*.



☐ Keyer Output 2

- ☐ R12, 18 - 1kΩ - (Brown-Black-Red)
- ☐ C3, 5 - 10nF (103)
- ☐ Q1, Q3 - 2N7000 -
- ☐ JP2 - 1x2 2.54mm header with jumper
- ☐ JX - 3.5mm TRS audio jack

Keyer output with PTT. If you do not wish to be able to enable / disable PTT, you can permanently jumper JP2 with bus wire, a scrap resistor lead, or similar. Transmit key lines, and PTT line are configured by *"#define tx_key_line_2"* and *"#define ptt_tx_2"* respectively, in *"keyer_pin_settings_k5bcq.h"*.



☐ Audio Output

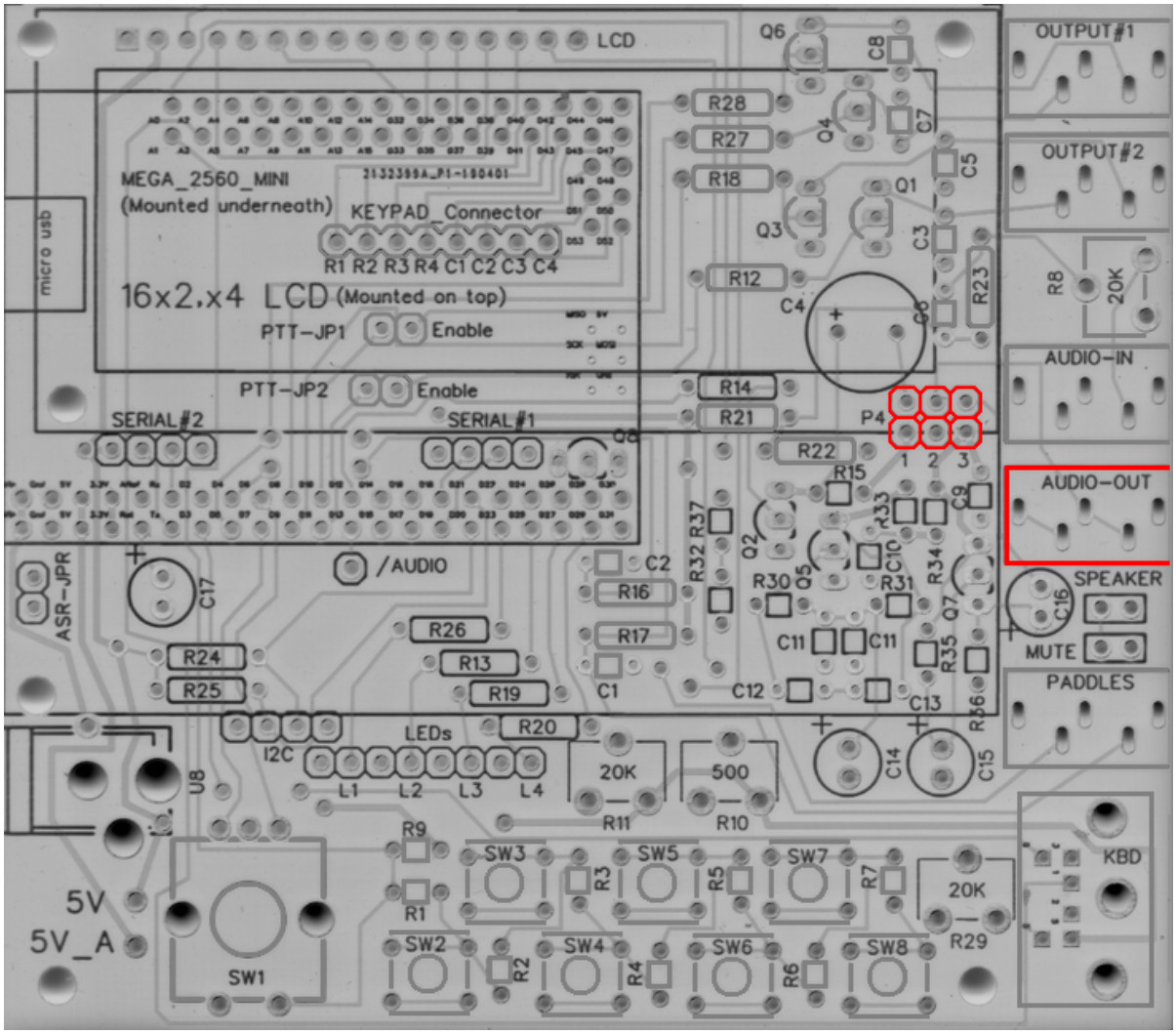
- ☐ P4A, P4B - 2, 1x3 or 1, 2x3 2.54mm headers with jumpers
- ☐ JX - 3.5mm TRS audio jack

This jumper block must be installed if EITHER audio module is to be included. This allows you to select between the Twin T Oscillator circuit, and the arudino's integrated Square wave output. You may choose to permanently jumper this if you do not wish to be able to switch. If you wish to use the TWIN T oscillator, install those parts, and place the jumpers on pins 2 & 3 of the headers (right 2), for both P4A and P4B. If you wish to use the square wave output, install those parts, and place the jumpers on pins 1 & 2 of the headers (left 2), for both P4A and P4B.

The audio is sent out to JX from both audio modules - no configuration changes are required, no matter which audio option you select.

The sketch defaults to the much more pleasant sounding, 600Hz Twin T oscillator. No configuration changes are necessary. To enable the Pulse-Width-Modulation square wave output, you must comment out *"#define OPTION_SIDETONE_DIGITAL_OUTPUT_NO_SQUARE_WAVE"* in *"keyer_features_and_options.h"* (it causes high/low logic instead of the PWM). After doing so, you **must** also change the pin sending the audio from 31, to 12. This is found under *"#define sidetone_line"* in *"keyer_pin_settings_k5bcq.h"*.

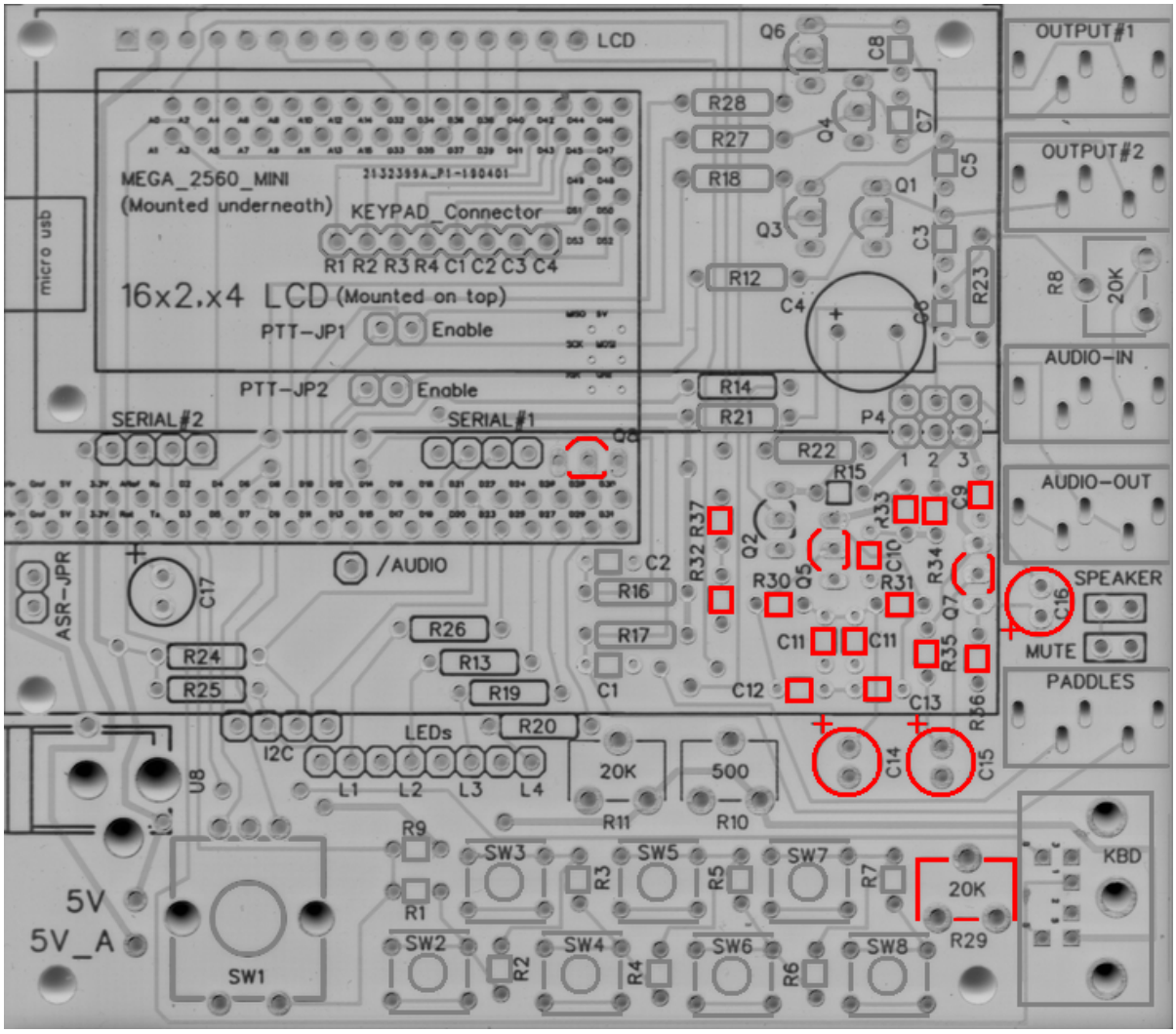
The sidetone can be toggled between "On", "Paddles Only", and "Off" using `"O"` in command mode (and `\O` in the command line interface). This setting is stored between powerdowns. It functions with both Twin T and Square wave output.


LCD
OUTPUT#1
OUTPUT#2
MEGA_2560_MINI
(Mounted underneath)
KEYPAD_Connector
R1 R2 R3 R4 C1 C2 C3 C4
16x2.x4 LCD (Mounted on top)
PTT-JP1
Enable
PTT-JP2
Enable
micro usb
SERIAL#2
SERIAL#1
AUDIO-IN
AUDIO-OUT
SPEAKER
MUTE
PADDLES
KBD
/AUDIO
ASR-JPR
I2C
LEDs
L1 L2 L3 L4
20K
500
5V
5V_A
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
R8
20K
R29
P4
1 2 3

☐ Audio Output (Twin T Oscillator)

- ☐ R29 - 20kΩ trimpot - (204)
- ☐ R30, 31, 33 - 3.3kΩ - (Orange-Orange-Red)
- ☐ R32 - 560Ω - (Green-Blue-Brown)
- ☐ R34 - 10Ω - (Brown-Black-Black)
- ☐ R35 - 150kΩ - (Brown-Green-Yellow)
- ☐ R36 - 100Ω - (Brown-Black-Brown)
- ☐ R37 - 100kΩ - (Brown-Black-Yellow)
- ☐ C9, 10, 11a, 11b, 12, 13 - 100nF - (104)
- ☐ C14, 15, 16 - 10uF Electrolytic
- ☐ Q5, 7 - 2N3904
- ☐ Q8 - 2N7000

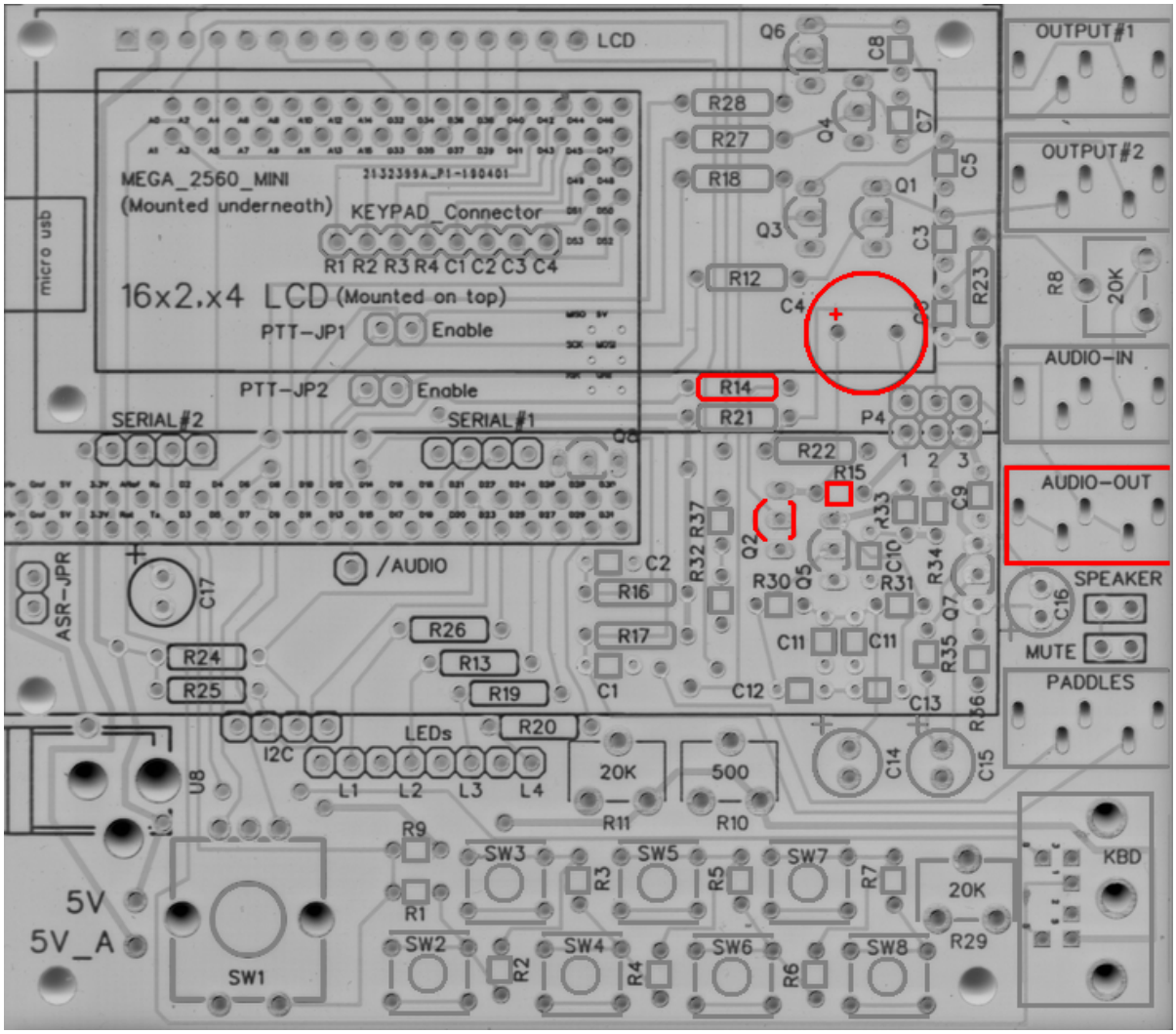
The audio output of this circuit generates a pure sine wave at a hardware-defined frequency, determined by the values of the components in the circuit, and these values have been configured for 600Hz. These values can be readily modified to change the desired frequency - but that is beyond the scope of this document. The circuit uses the high/low logic from Pin 31, and inverts it with Q8. The advantage is the audio quality - the disadvantage of this circuit is the inability to modify the sidetone without replacing the parts.



☐ Audio Output (Square Wave)

- ☐ R14 - 100Ω - (Brown-Black-Brown)
- ☐ R15 - 220Ω - (Red-Red-Brown)
- ☐ C4 - 100uF - Electrolytic
- ☐ Q2 - 2N3904
- ☐ JX - 3.5mm TRS audio jack

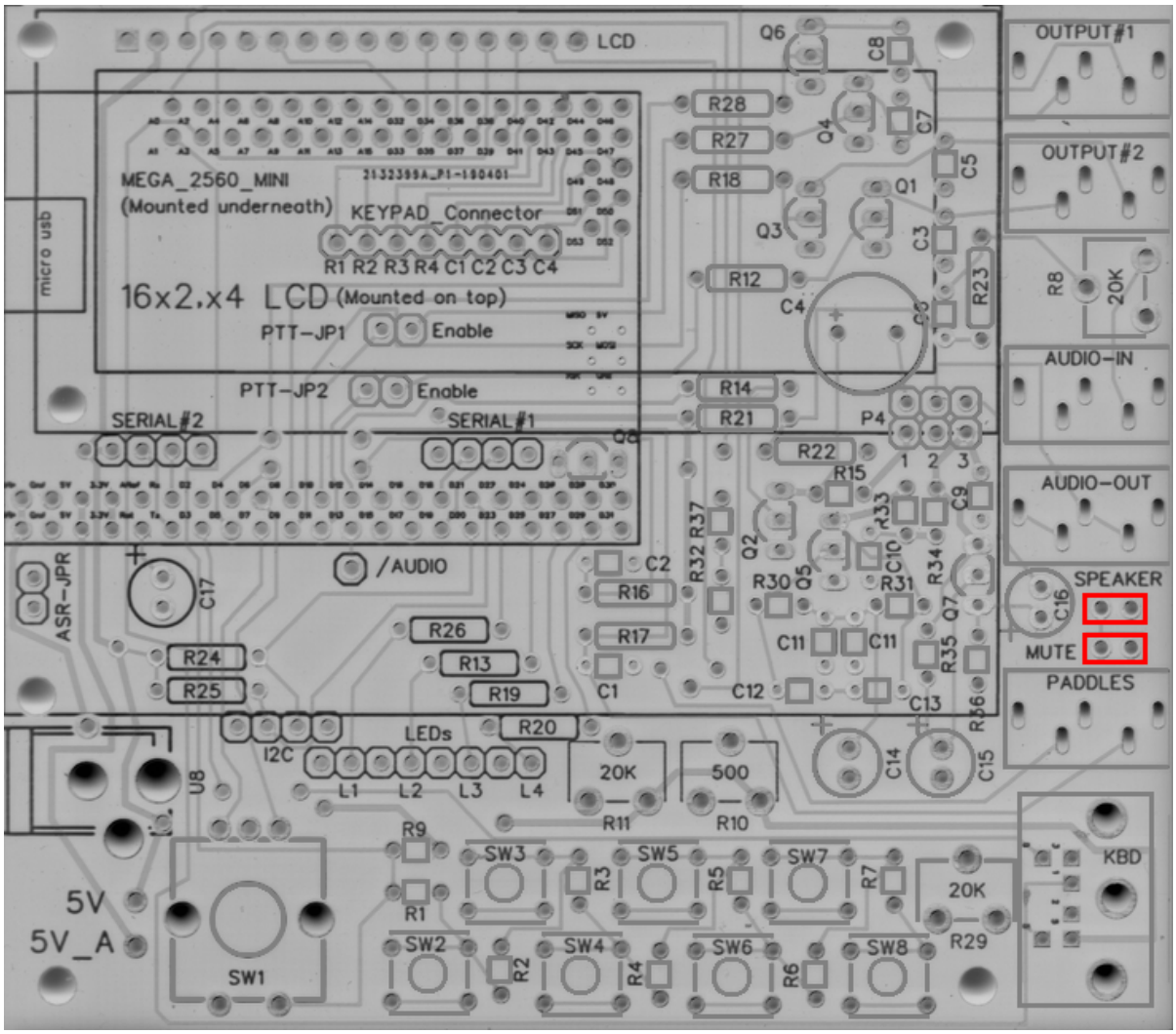
The audio output using the Square Wave output is a tone generated by Pulse-Width Modulation. While not nearly as nice sounding as the Twin T, it has a number of advantages. Its tone can be adjusted using `"F"` in the command mode (and `\f ###` in the command line interface), to the users preference at any time. The setting is saved between powerdowns. The default value (i.e. first boot and factory reset) is set by *"#define initial_sidetone_freq"* in *"keyer_settings_k5bcq.h"*. The limits (low and high) are set by *"#define sidetone_hz_limit_low"* and *"#define sidetone_hz_limit_high"* respectively, in the same file.



☐ Speaker Output

- ☐ SP1 - Speaker
- ☐ P2 - 1x2 2.54mm male header

SP1 is hookups for an optional, off-board speaker. Audio output is taken from both Twin T and Square Wave outputs, so no modification is necessary. The circuit is designed to mute if Audio Output plug is in use. P2 is an optional jumper, that allows you to detach the speaker from ground, muting the speaker output. (jumpered = on, unjumpered = muted)



3.4 Displays / Indicators

There are number of inexpensive LCD displays that are available from a number of sources. There are too many to list individually, but suffice it to say, if it uses serial or i2c, you can hook it up.

There are multiple settings for displays. The assigned pins are set by *"defined(FEATURE_LCD_4BIT)"* in *"keyer_pin_settings_k5bcq.h"* for serial displays. i2c is configured by enabling the appropriate i2c option in *"keyer_features_and_options.h",* based on the hardware you are using. The k5bcq configuration comes defaulted to a serial, 1602 display. To adjust the LCD display size (serial or i2c - 1604, 2004, etc), change *"define LCD_COLUMNS"* and *"define LCD_ROWS"* to their appropriate values in *"keyer_settings_k5bcq.h".*

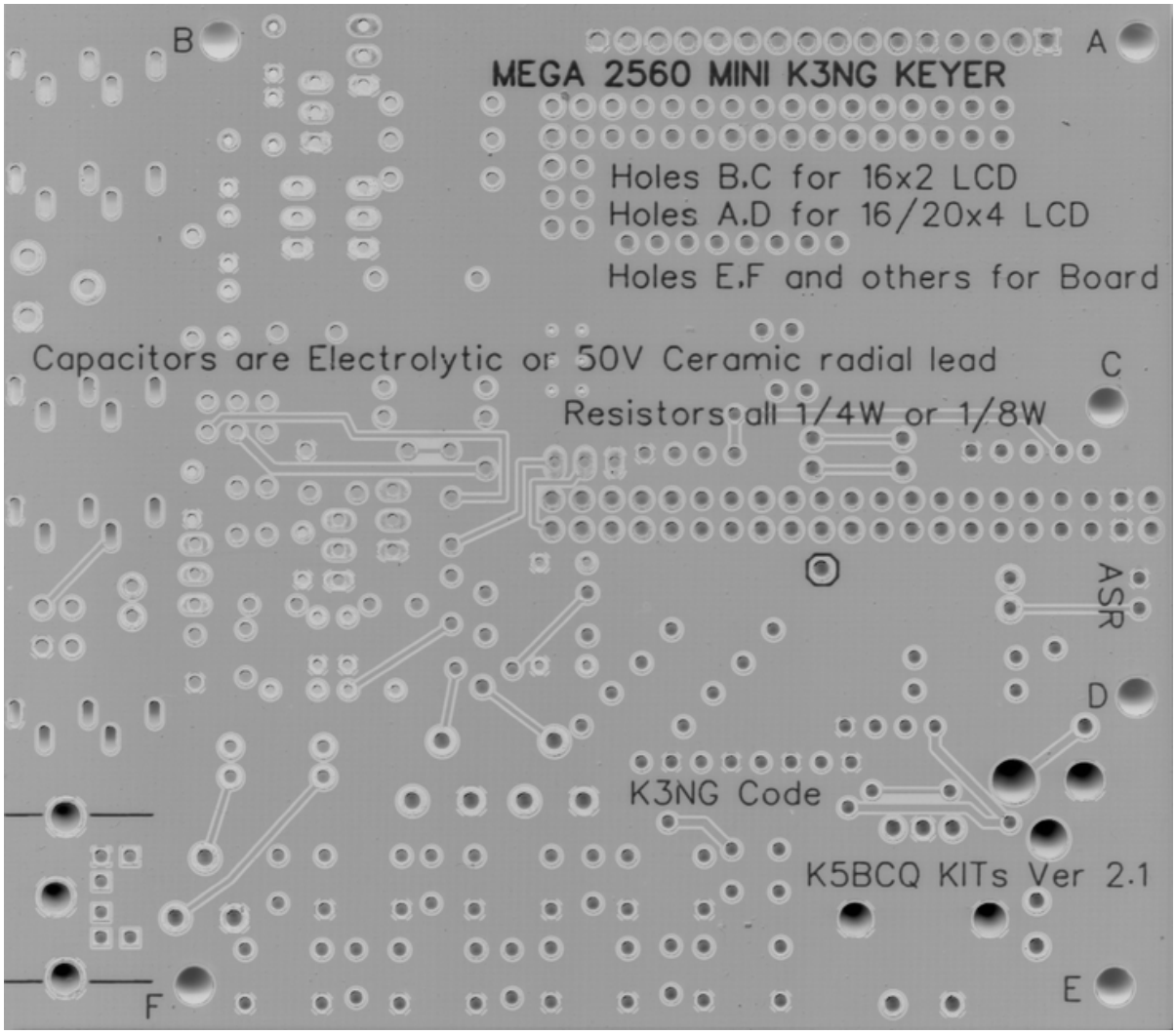
☐ LCD Display (Serial)

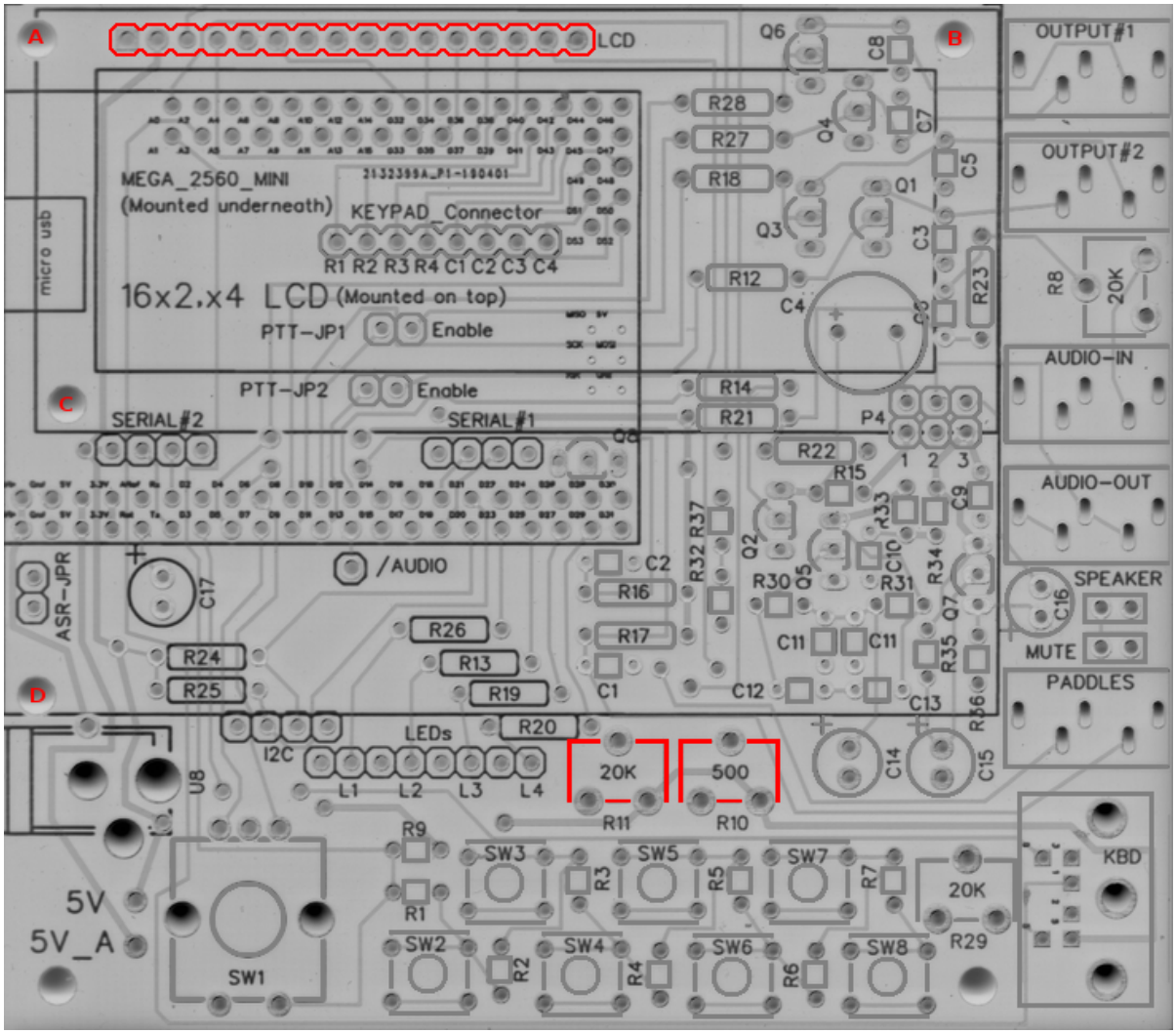
- ☐ R10 - 500Ω - 6mm top-adjust trimpot
- ☐ R11 - 20kΩ - 6mm top-adjust trimpot
- ☐ U1 - LCD display (1602, 1604, 2004)
- ☐ 1x16 2.54mm male & female headers

  Description of the installation of a display is relatively *difficult.* Not because the process is difficult, but due to the variety. Suffice it to say, the description is much more difficult than the execution. To install a serial display (1602, 1604, 2004), connect them via the 16 pin "LCD" headers. To install attached to the board, use the same process as the installation of the Arduino - cut appropriate lengths (1x16) male and female headers, and lightly combine the headers. Sandwich them between the display and the PCB, and solder the ends. Once you've made sure they're flush and perpendicular with their respective boards, finish soldering the pins. Install R10 and R11 to control the display's backlight and contrast, respectively.

  Builders note: Installing spacers on the display and using them to ensure level installation is highly recommended. It makes the entire job easier, and results in a neater product - even if you have no intention of leaving them in place. Mounting holes have been added to the board to affix the display if you so choose. (Also recommended)

Builders note: Adjust the backlight to the desired level first, then adjust contrast until just after the blocks disappear on the display. This will provide the crispest looking display.


B
A
MEGA 2560 MINI K3NG KEYER
Holes B.C for 16x2 LCD
Holes A.D for 16/20x4 LCD
Holes E.F and others for Board
Capacitors are Electrolytic or 50V Ceramic radial lead
C
Resistors all 1/4W or 1/8W
ASR
D
K3NG Code
K5BCQ KITs Ver 2.1
F
E

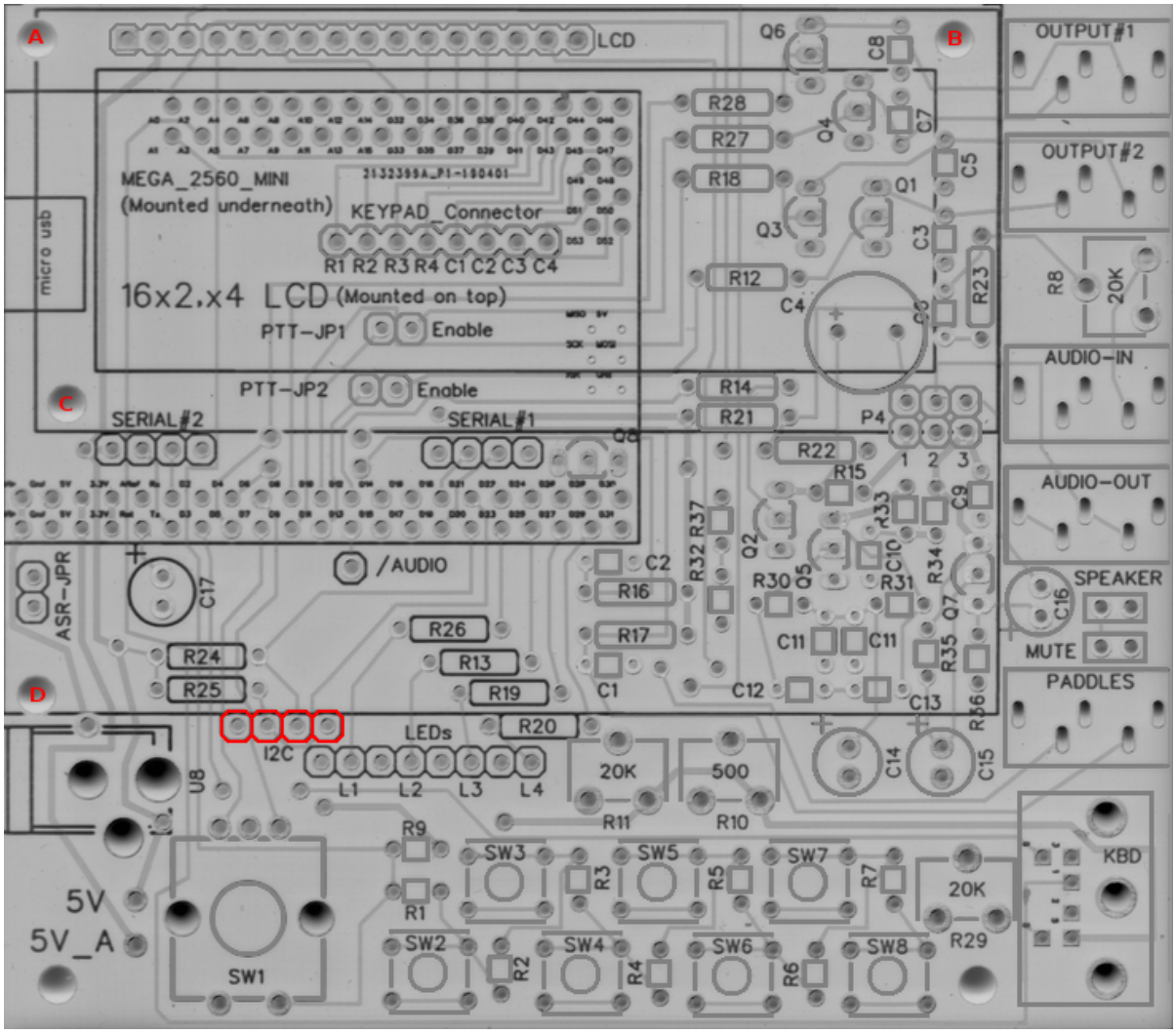

A
B
C
D
LCD
Q6
C8
OUTPUT#1
OUTPUT#2
R28
R27
R18
Q4
C7
C5
Q1
Q3
C3
MEGA_2560_MINI
(Mounted underneath)
KEYPAD_Connector
R1 R2 R3 R4 C1 C2 C3 C4
micro usb
16x2.x4 LCD (Mounted on top)
PTT-JP1
Enable
R12
C4
C6
R23
R8
20K
AUDIO-IN
PTT-JP2
Enable
R14
R21
P4
SERIAL#2
SERIAL#1
Q8
R22
R15
1 2 3
C9
AUDIO-OUT
R33
R37
R32
Q2
C10
R34
/AUDIO
C2
R16
R30
Q5
R31
Q7
SPEAKER
C16
ASR-JPR
C17
R26
R17
C11
C11
R35
MUTE
R24
R13
C1
C12
C13
R56
PADDLES
R25
R19
R20
LEDs
I2C
20K
500
C14
C15
U8
L1 L2 L3 L4
R11
R10
R9
SW3
SW5
SW7
R3
R5
R7
20K
KBD
5V
5V_A
R1
SW2
SW4
SW6
SW8
R29
SW1
R2
R4
R6

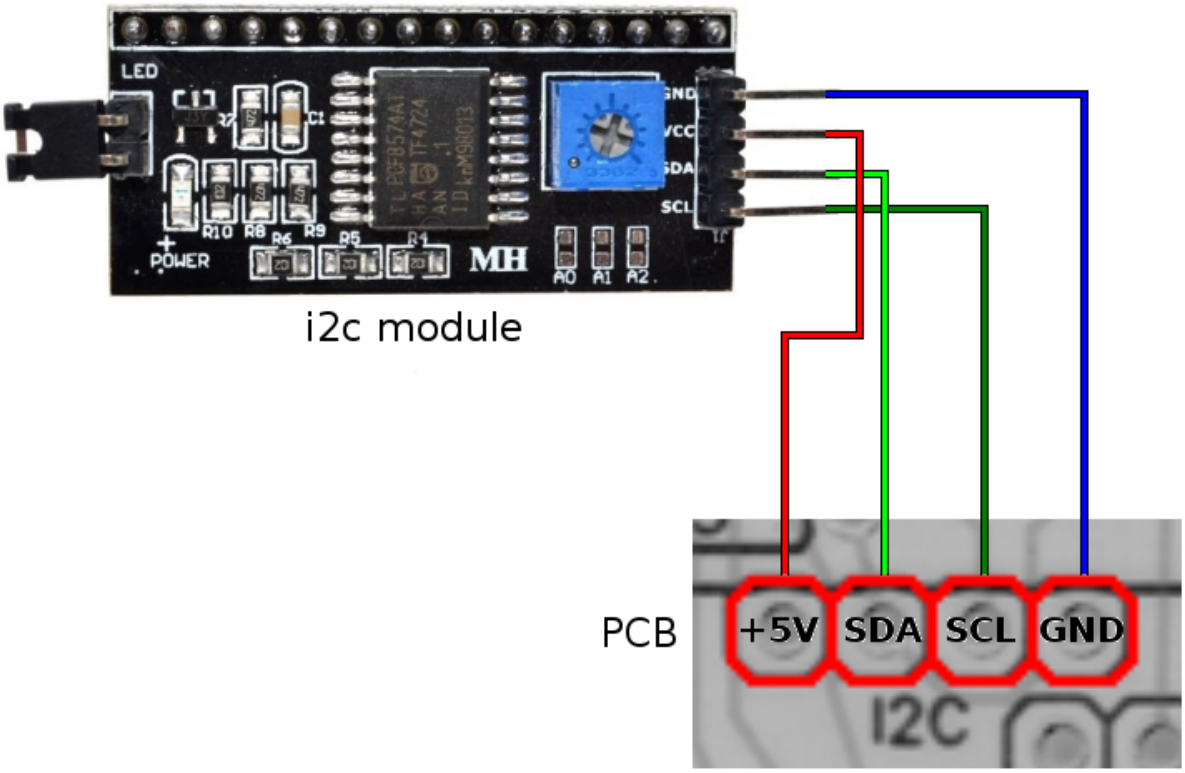
☐ I2C Display

- ☐ U1 - LCD display (1602, 1604, 2004)
- ☐ 1x4 2.54mm male & female headers
- ☐ Dupont wires or similar

The installation of an i2c display is similar to that of a standard serial display, but is complicated by the location of the i2c headers and device pinouts. You will likely need to off-board the i2c display. Individual configurations are beyond the scope of this document, but suffice to say - install headers and/or jumper wires and connect as labeled.

Builders note: Left to right, the i2c pins are: +5volts, SDA, SCL, Ground. You may also need to adjust the i2c device address in the main keyer file (k3ng_keyer.ino), if you are having difficulty connecting using some of the i2c hardware. This is rare, but possible - particularly when using inexpensive Chinese clones.



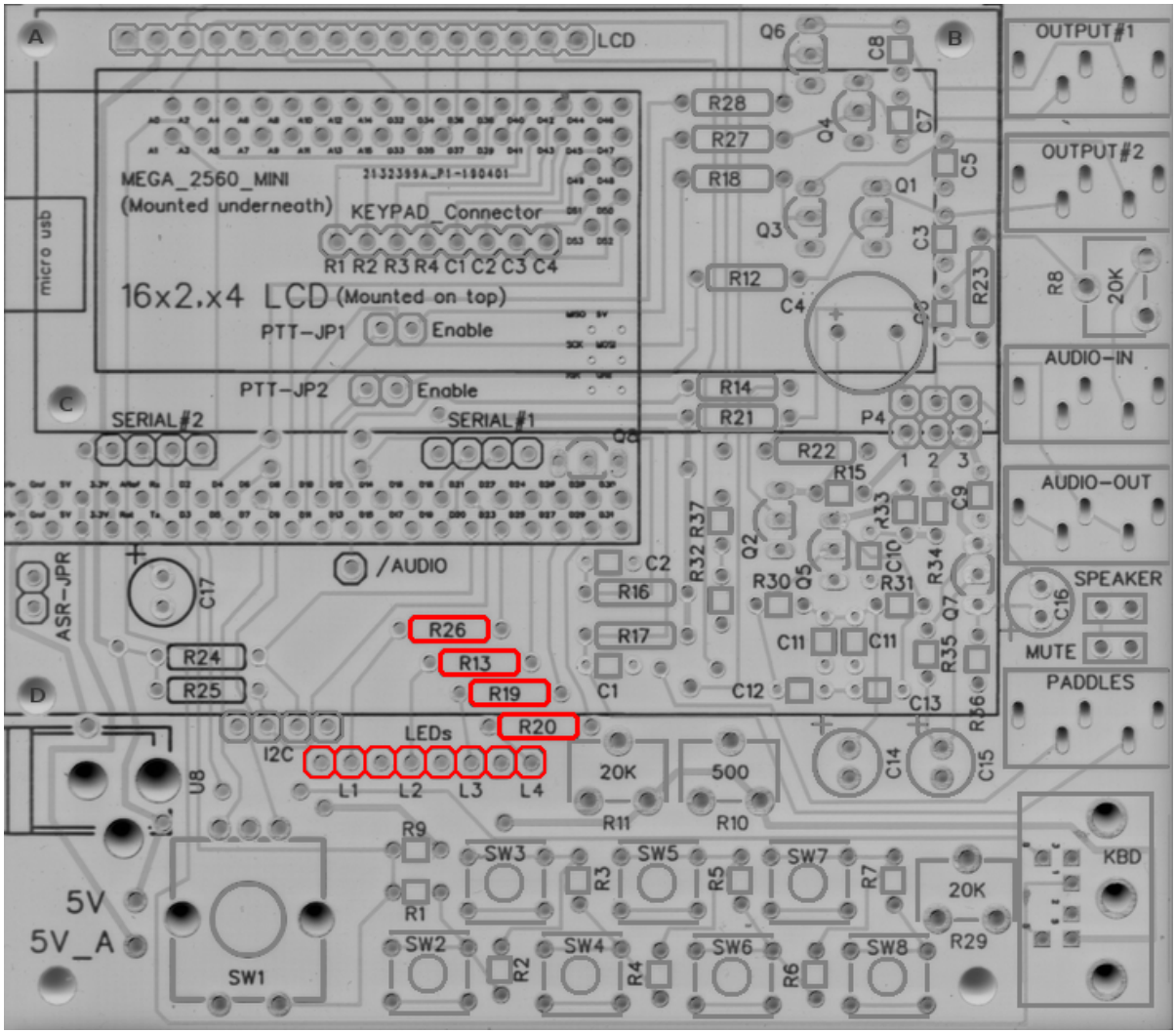

LED
GND
VCC
SDA
SCL
POWER
MH
A0
A1
A2
i2c module
PCB
+5V
SDA
SCL
GND
I2C

☐ LED Indicators

- ☐ R13, 19, 20, 26 - 220Ω - (Red-Red-Brown)
- ☐ LED1, 2, 3, 4 - LEDs of your color choice

LED indicators for command mode, audio input for Goertzel decoder, and Right / Wrong for sending practice. These can easily be modified for different purposes, if you like. Depending on your enclosure design, you may wish to add headers instead of the LEDs themselves, so you can readily off-board them to elsewhere on your enclosure. See *"keyer_pin_settings_k5bcq.h"* for changing pin assignments. The sketch's default settings are:

LED	Pin	Default Setting
LED 1	23	CW Decoder Indicator
LED 2	25	Send Practice Wrong
LED 3	27	Send Practice Correct
LED 4	29	Command Mode Indicator

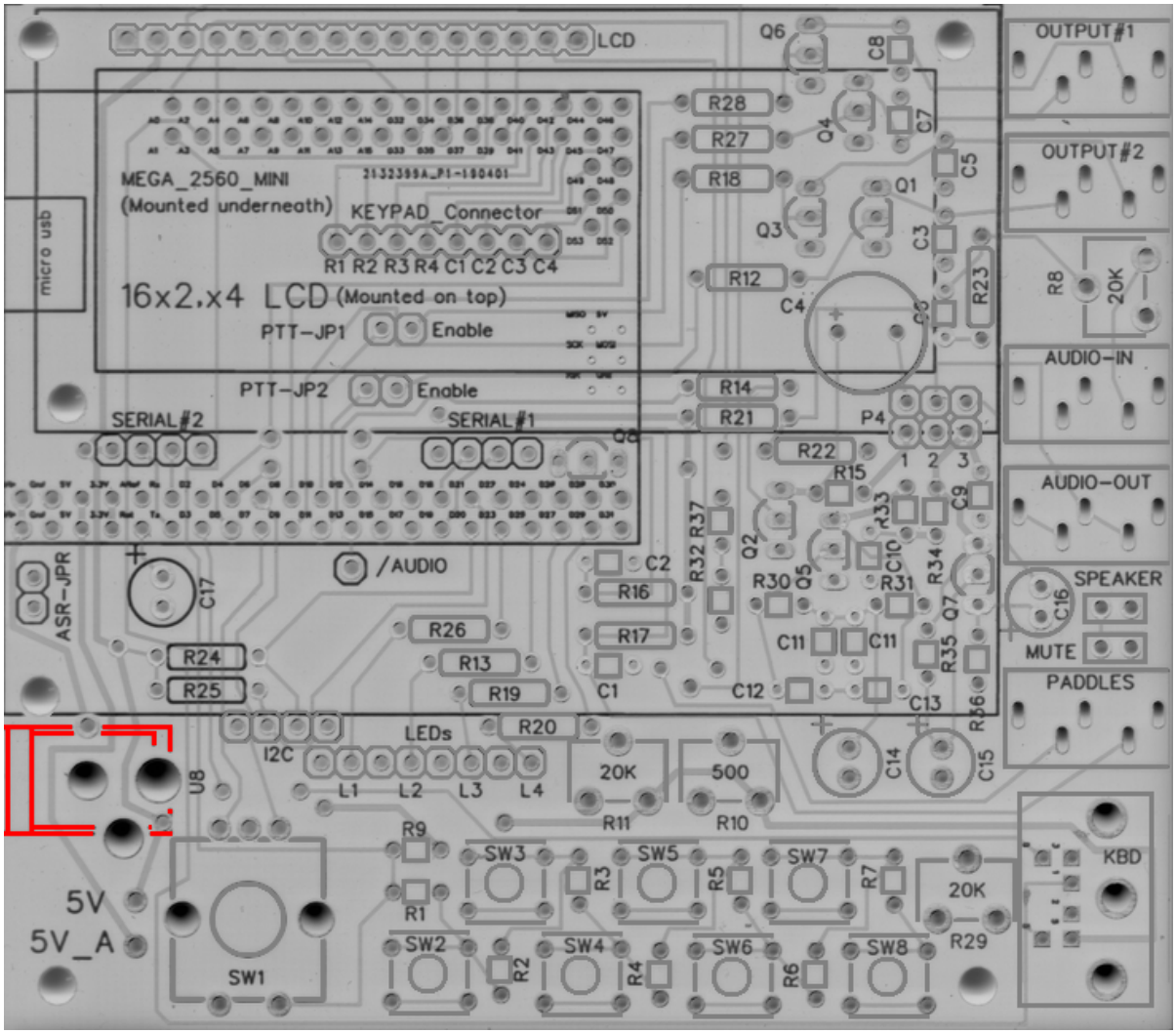


3.5 Miscellaneous

☐ DC Power Jack

☐ J6 - 2.1x5.5mm DC Barrel Socket

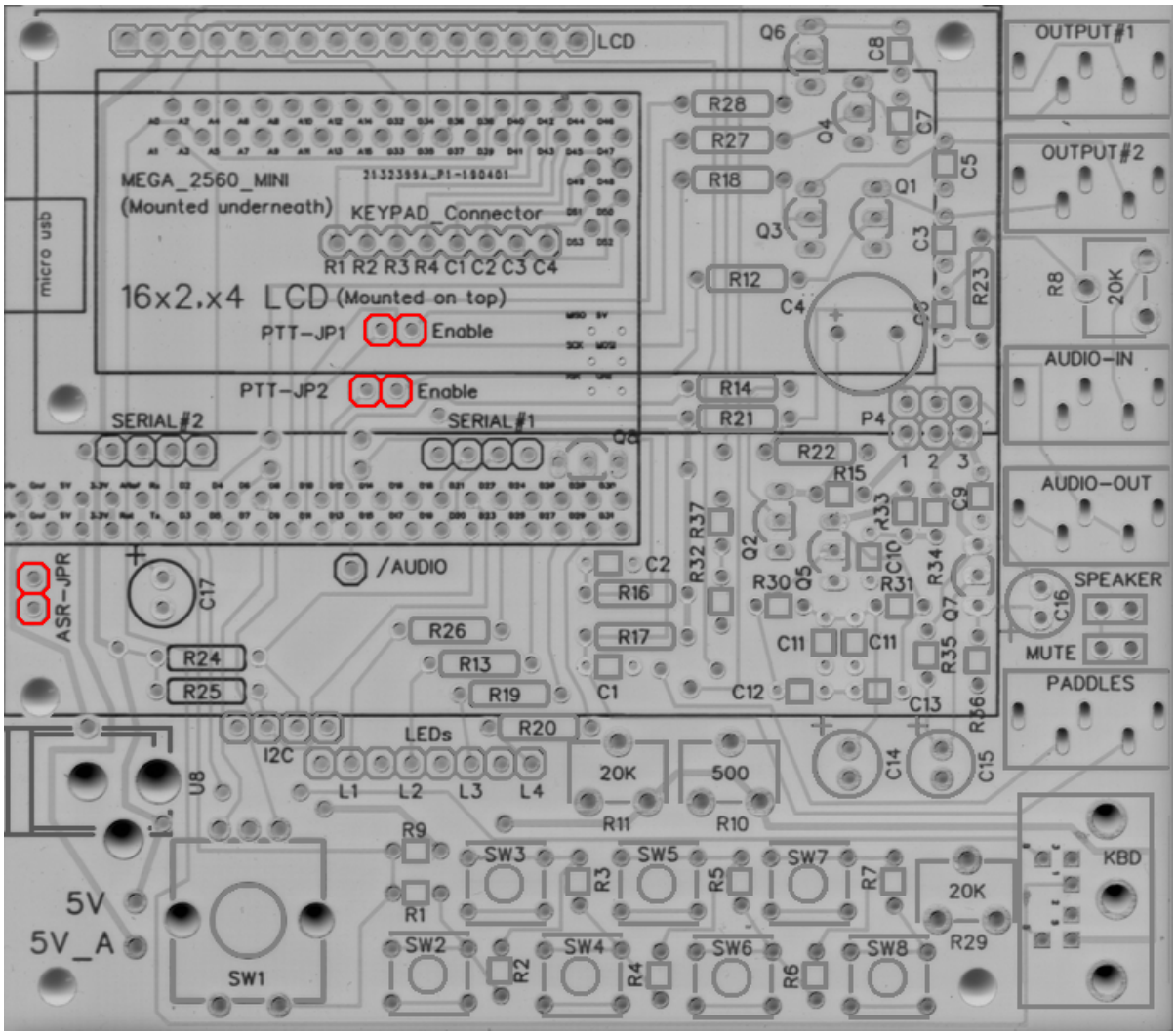
This is DC input of 7-12v. It is routed to the Voltage In lines on the Arduino, and takes advantage of its linear voltage regulator to output 5v to all of the other parts. Note: a battery or other voltage in can readily be installed using the Pad just above the socket's ground.



☐ ASR Jumper, PTT Jumper TX_1, PTT Jumper TX_2

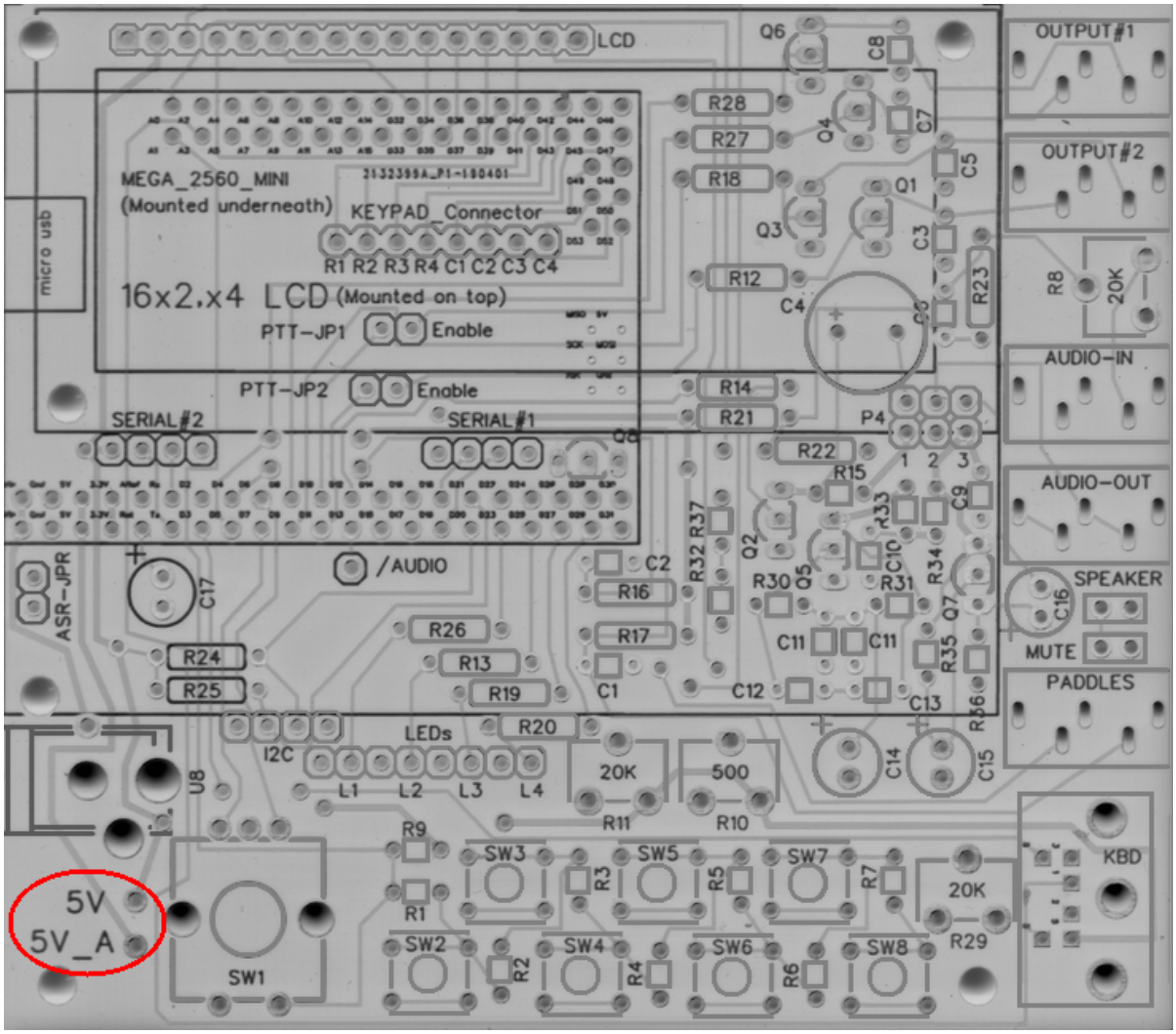
☐ 1x2 - 2.54mm male header and jumper (each option)

You can adjust these features by jumpering these connections with bus wire or scrap resistor leads. However, it's likely a better idea to install 1x2 2.54mm headers and jumpers. This allows you the ability to enable / disable them quickly and easily.



☐ Power Jumper

– This set of pads allows you to either power the peripherals using the 5 volt output of the arduino, or input 5v from an external source to power them. If you plan to use it independent of a computer, you likely will just permanently jumper this.



4 Flashing Instructions

The code for the K3NG keyer, in any of its designs (homebrew or otherwise!) resides on the K3NG Github. Goody has kindly including pull requests to enable hardware for this build into the K3NG main code, resulting in a very easy method for flashing the most recent code and features!

1. Download the code from K3NG Github.
2. Unpack (unzip) the arduino project wherever you like. The convention is in the $HOMEDIRfolder.
   - *If you git cloned the repo... you likely don't need me to explain the rest of this process to you...*
3. Make sure you have the code (containing the k3ng_keyer.ino and a bunch of .h files) in a folder name k3ng_keyer. If you do not, the Arduino IDE will complain, and likely move the file on you.
4. Copy the contents of the library folder (k3ng_cw_keyer/libraries) to your Arduino/libraries folder ("~/Arduino/libraries" on Linux).
   - *Depending on your method and location of extraction, you may just skip this step.*
5. Open the Sketch. The actual sketch file is *"k3ng_keyer.ino"*.
   - You can open it either by double-clicking on the file in your file explorer, or *"File → Open"* in the Arduino IDE.
6. Your path now splits. You can either:
   - Edit the keyer configuration files by hand to enable every feature you want, and assign the pins yourself (in which case you'll edit *"keyer_features_and_options.h", "keyer_pin_settings.h", "keyer_settings.h",* and possibly *"keyer_debug.h"*.
   - Enable a hardware configuration by un-commenting (deleting the `//` from the front of a line) a single line - *"#define HARDWARE_K5BCQ"* from *"keyer_hardware.h"*.
7. Since we're going to assume you chose the second option, you *should* be good to go. This configuration has all of the pin settings already performed, and all non-invasive options turned on. If you want to enable or disable anything, now is the time. Features can be enabled by uncommenting them in *"keyer_features_and_options_k5bcq.h"*, and can be disabled by commenting them back out. Pin settings can be adjusted as well. be sure to edit them as described in the code (and in the above build section).
   - **REMEMBER** this configuration comes with the Twin T oscillator as default. If you are using the square wave output, please adjust the values noted in the code (and documented in the build section).
8. Connect the Arduino to your computer using the USB cable.
   - If this is done with the Arduino connected to the PCB, be sure to **remove** (unjumper) the ASR jumper. It won't work for you if you don't. Reinstall when you're done.
9. Set up your board in the Arduino IDE.
   - The Arduino Mega 2560 Pro Mini has the same pinouts, chip, and specs as the full-size Mega. Thus, no additional boards need to be installed. To select it: Tools → Board → Arduino AVR Boards → "Arduino/Genuino Mega or Mega2560"

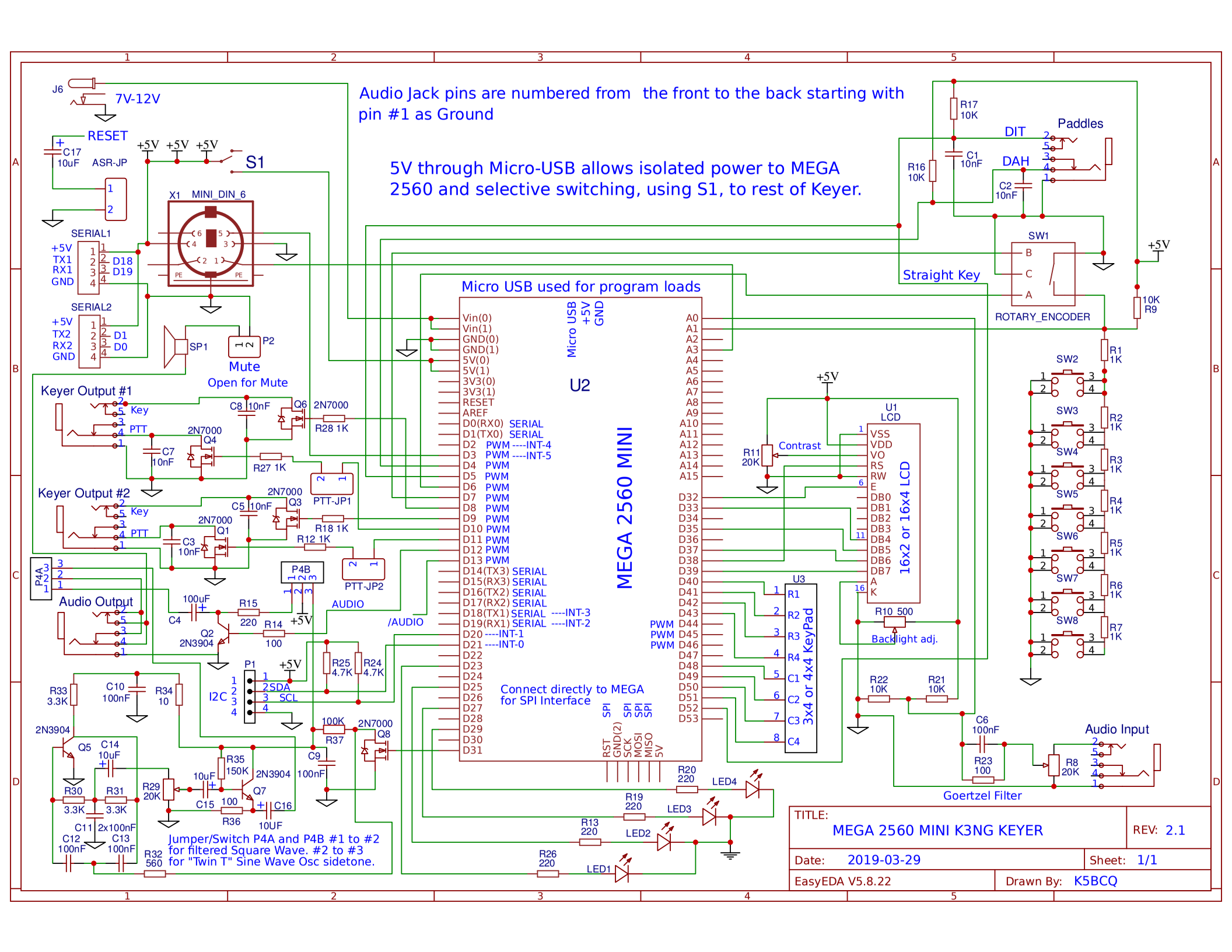
- You shouldn't need to adjust it, but make sure that the correct processor is selected. To check: Tools → Processor → "ATmega2560 (Mega2560)"
- Select the appropriate port. You likely will only have one available. To select it: Tools → Port → Whichever is available
- You shouldn't need to adjust it, but make sure you have the correct programmer selected. To check: Tools → Programmer → "AVRISP mkII". Others will likely work, but this default has always worked for me.

10. Click "Upload"

    - This will automatically compile the code, save it, and transmit it to the attached Arduino
    - You can safely ignore the error regarding the "K3NG_PSKeyboard" library being the wrong category. It's just a warning.

11. Wait patiently. The message window at the bottom will keep you updated on how it's progressing (compile, upload, errors). When it indicates that the upload is finished, you can disconnect the Arduino, and your keyer is ready to go!

5 Schematic


Audio Jack pins are numbered from the front to the back starting with pin #1 as Ground
5V through Micro-USB allows isolated power to MEGA 2560 and selective switching, using S1, to rest of Keyer.
Micro USB used for program loads
MEGA 2560 MINI
U2
Connect directly to MEGA for SPI Interface
Straight Key
Paddles
DIT
DAH
ROTARY_ENCODER
Keyer Output #1
Keyer Output #2
Audio Output
Audio Input
Mute
Open for Mute
Contrast
Backlight adj.
16x2 or 16x4 LCD
3x4 or 4x4 KeyPad
Goertzel Filter
Jumper/Switch P4A and P4B #1 to #2 for filtered Square Wave. #2 to #3 for "Twin T" Sine Wave Osc sidetone.
TITLE: MEGA 2560 MINI K3NG KEYER
REV: 2.1
Date: 2019-03-29
Sheet: 1/1
EasyEDA V5.8.22
Drawn By: K5BCQ