K5BCQ K3NG Keyer v0.1

woodjrx

May 19, 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!

2 Bill of Materials

Reference	Value	Quantity	Part #	Notes
C1				
R1				
R10	500	1		6mm top-adjust.
R8, 11, 29	20k	3		6mm top-adjust.
Q1, 3, 4, 6, 8	2N7000	5		
Q2, 5, 7	2N3904	3		
LED1, 2, 3, 4		4		LED of your choice of colors
SW1		1		
SW2, 3, 4, 5, 6, 7, 8		7		
J1, 2, 3, 4, 5		5		
J6		1		DC plug
X1		1		MiniDin 6
SP1		1		Off-board speaker
U1	Display	1		1602, 1604, 2004 Serial LCD
U2	Mega2560	1		Mini variant
U3	Keypad	1		3x3 or 3x4 keypad
P1, 2, 4A, 4P, ASR-JP				2.54mm Male headers

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

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

- ☐ R9 - 10kΩ - (Brown-Black-Orange)
- ☐ R1, 2, 3, 4, 5, 6, 7 - 1kΩ - (Brown-Black-Red)
- ☐ SW2, 3, 4, 5, 6, 7, 8 - 6mm x 6mm 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"*.

☐ Goertzel Audio Decoder

- ☐ R21, 22 - 10kΩ - (Brown-Black-Orange)
- ☐ R23 - 100Ω - (Brown-Black-Brown)

- ☐ R8 - 20kΩ Trimpot - (203)
- ☐ 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)
  - ☐ Q4, Q6 - 2N7000 -
  - ☐ C7, 8 - 10nF (103)
  - ☐ 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)
  - ☐ Q1, Q3 - 2N7000 -
  - ☐ C3, 5 - 10nF (103)
  - ☐ 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.

☐ 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

☐ LCD Display (Serial)

- R10 - 500Ω - 6mm top-adjust trimpot
- R11 - 20kΩ - 6mm top-adjust trimpot
- U1 - LCD display (1602, 1604, 2004)

☐ I2C Display

☐ LED Indicators

☐ R13, 19, 20, 26 - 220Ω - (Red-Red-Brown)

☐ LED1, 2, 3, 4 - LEDs of your color choice

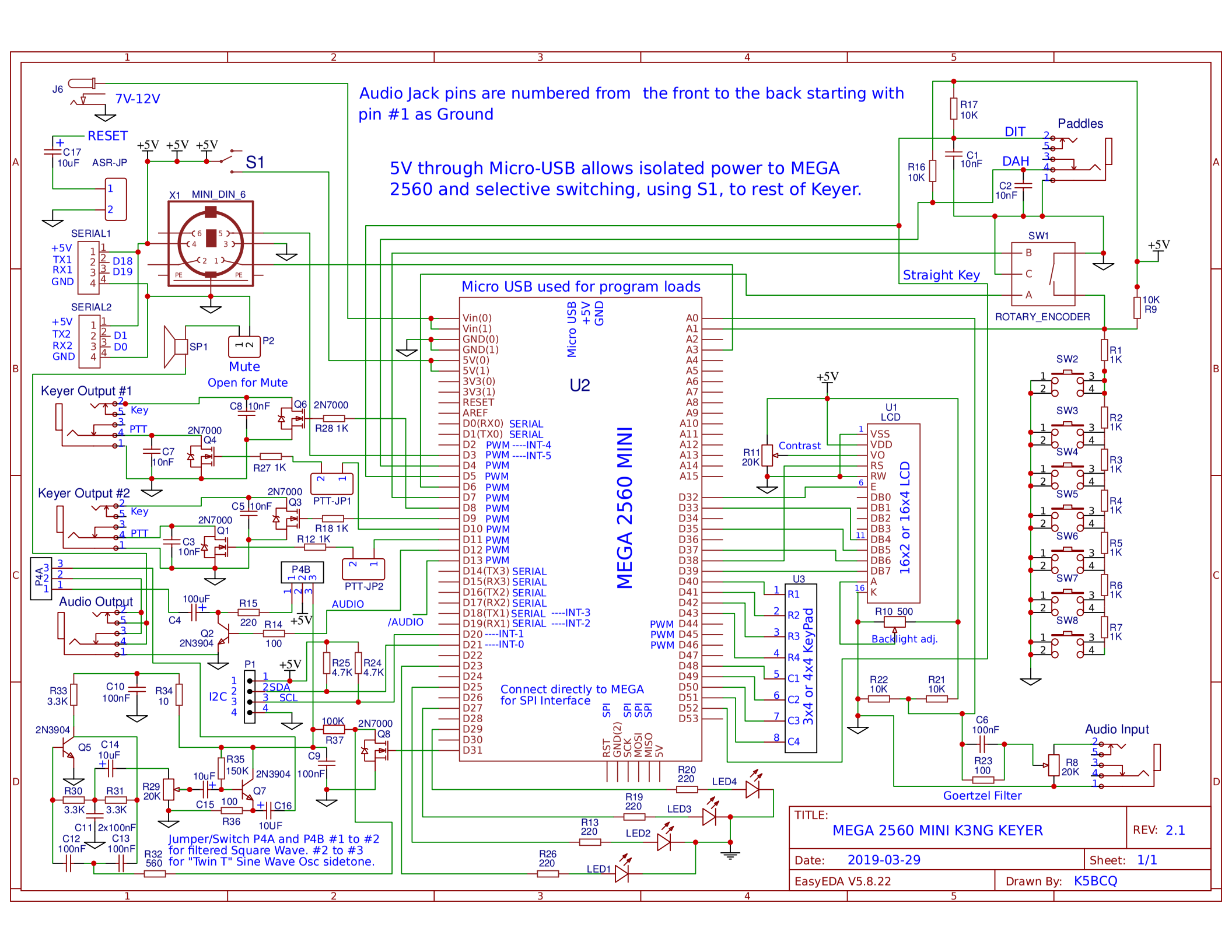
LED indicators for command mode, audio input for Goertzel decoder, and Right / Wrong for sending practice. These can easily be modified for different purposes, if you like. Depending on your enclosure design, you may wish to add headers instead of the LEDs themselves, so you can readily off-board them 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

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!

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