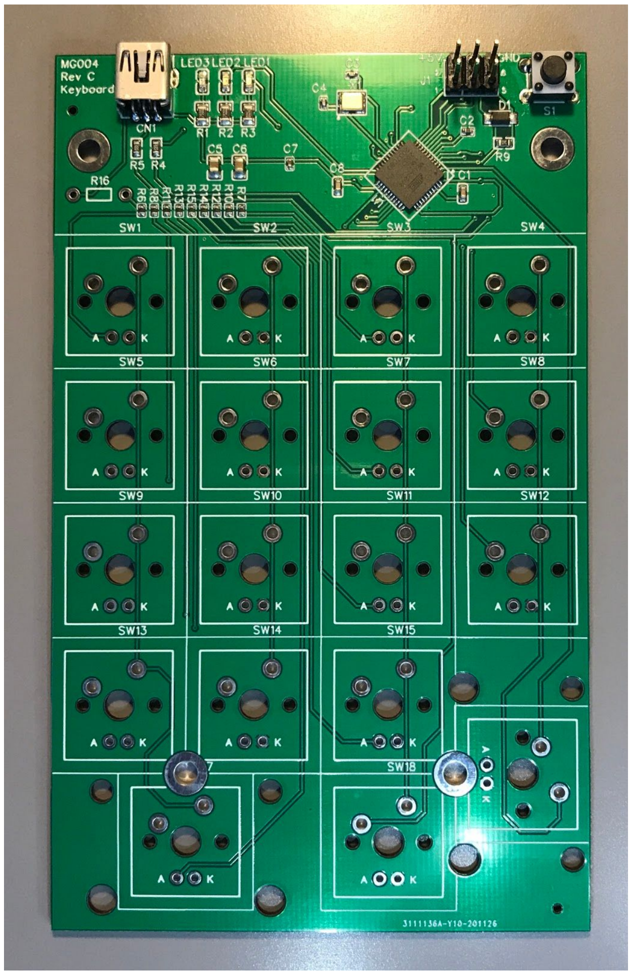
MG004 USB Mechanical Numeric Keyboard

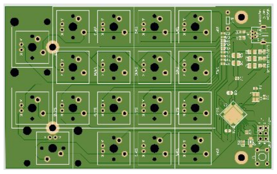
What is it?

MG004 is a USB numeric keyboard designed for Cherry mechanical switches (or third party equivalents with the same form factor). It is based on the ATMEGA32U4 microcontroller, which can be reprogrammed (using Arduino IDE) to accommodate different keys.

Switches, keycaps and LEDs are not supplied, as there are numerous types to choose from depending on the user's preference.



What’s in the kit?

Name	Quantity	Description	Picture	Present?
PCB	1	MG004 assembled and programmed PCB		☐

What’s not supplied?

Name	Quantity	Description
Cherry Switches	18	Numerous types are available with different characteristics. Must be suffixed “NW” (no LED, but with PCB mounting pins)
2u Stabiliser Plate	2	For Enter and “0” keys. Cherry PN G99-0742. Can be found on eBay
Keycaps	18	
3mm LEDs (optional)	18	Different colours and brightnesses are available
R16 (optional)	1	Wire link or resistor for LEDs
Mini USB cable	1	To connect to host (Mini USB to appropriate connector for host)
Case (optional)	1	Home designed case, or if you like the minimalistic look, some stick-on feet to go underneath the PCB

How do I build it?

This kit assumes basic through hole soldering skills. If you’re not too confident, I recommend searching for an online tutorial, there are some good ones on YouTube.

Recommended tools include:

- Soldering iron (ideally temperature controlled)
- Multicore solder
- Small snips to cut off leads
- Small pliers

- Desoldering pump and/or braid
- Anti-static wrist strap (or steer clear of materials that cause static and touch a grounded object every now and then).

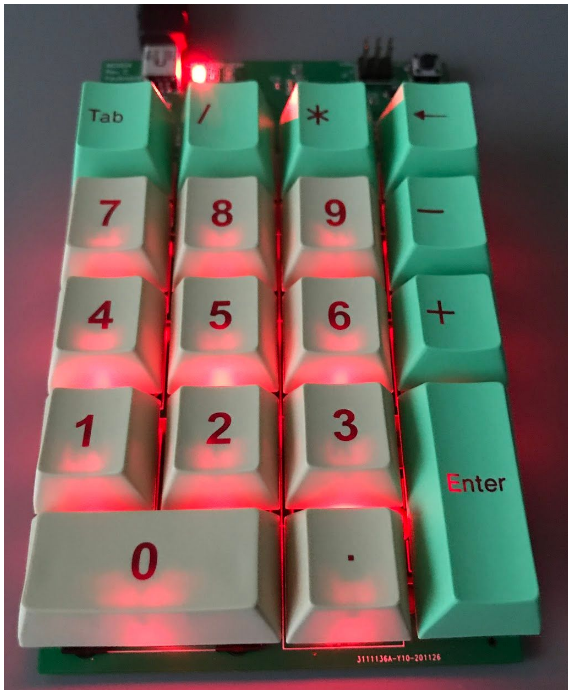
Assembly is a relatively simple case of soldering in 18 switches, and then adding two stabiliser plates (not essential, but makes 2u keys work better) and keycaps.

If backlighting is needed, LEDs need to be fitted through the switches and soldered ahead of fitting keycaps.

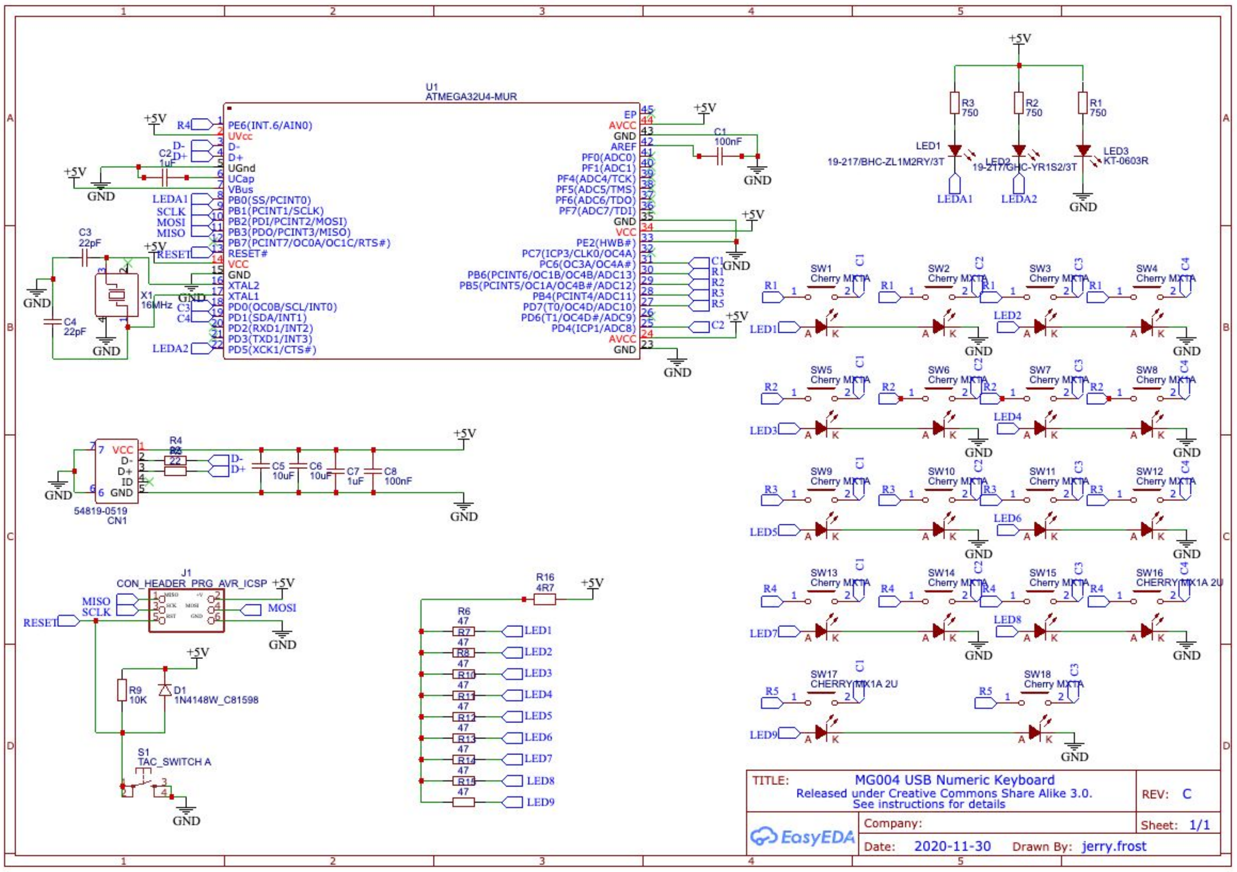
If a wire link (a cut off LED lead will work) is used for R16, then current through the LEDs will be about 20mA. This can be reduced, if required, by using a resistor, 4.7 Ohms will provide about 10mA to each LED.

If you have flux cleaner, clean all joints. Now inspect them carefully for issues (a magnifying glass of some sort can be very helpful, the camera on some phones works quite well).

Below is a picture of the finished article with switches, keycaps and LEDs fitted:



How does it work?



U1 does the hard work, handling all the USB keyboard interface with the host, and scanning the keys for keypresses. CN1 is the USB connector (power and data) and J1 is a header for an In-System-Programmer (more on this later).

LED1 is the receive indicator, LED2 transmit, and LED3 power.

U1 uses the Arduino “keyboard” and “keypad” libraries to appear as a keyboard to the host, and scan the keypad, respectively. The keypad library does not need isolation diodes, and therefore none are fitted.

How do I use it?

Plug it into a computer and type away!

Tab	/	*	←
7	8	9	-
4	5	6	+
1	2	3	Enter
0			

The standard software loaded is set up for the above key pattern, and the code the MG004 sends for keypresses are as per the key legends above, except for:

- Tab - “\t”
- ← - “\b”
- Enter - “\n”

This code has worked fine for me, for both Windows and Mac hosts. A new sketch can be loaded over USB using the ARDUINO IDE if different key values or even a different mode of operation are required. Please make sure to tell the IDE that the board is a Sparkfun ProMicro, with a 5V ATMEGA 32U4. If the board gets totally bricked, then the ISP can be used to load a new bootloader.

Acknowledgements/Legal

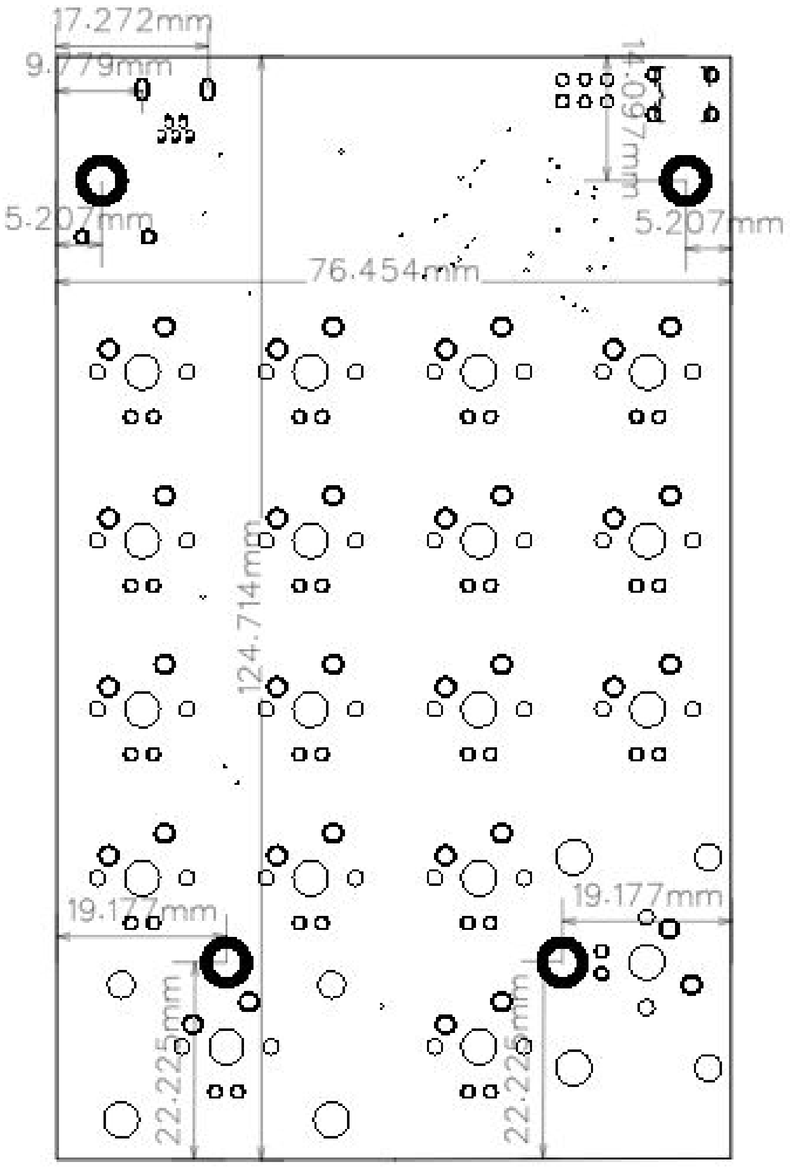
MG004 used the SparkFun Pro Micro as a starting point for development. The Pro Micro was developed by SparkFun Electronics, based on an original design by Team Arduino.

The Arduino code

This can be modified to change key layouts etc.

```
#include <Keypad.h>
#include <Keyboard.h>

const byte ROWS = 5;
const byte COLS = 4;

char hexaKeys[ROWS][COLS] = {
  {'\t', '/', '*', '\b'},
  {'7', '8', '9', '-'},
  {'4', '5', '6', '+'},
  {'1', '2', '3', '\n'},
  {'0', 'X', '.', 'Z'}
};

byte rowPins[ROWS] = {10, 9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);

void setup(){
  Keyboard.begin(); //Init keyboard emulation
}

void loop(){
  char customKey = customKeypad.getKey();

  if (customKey){
    Keyboard.write(customKey);  // send character to the computer via Keyboard HID
    delay(100);  // delay to debounce
  }

}
```

Mounting Dimensions:
The following should help determine mounting hole locations:



The mounting holes are 3.2mm diameter, with the solder pad around them being 6.0mm diameter.

The space available for screw heads for the lower two holes may be limited, depending on the switch type fitted.