Reverse engineering a VEX IQ Controller

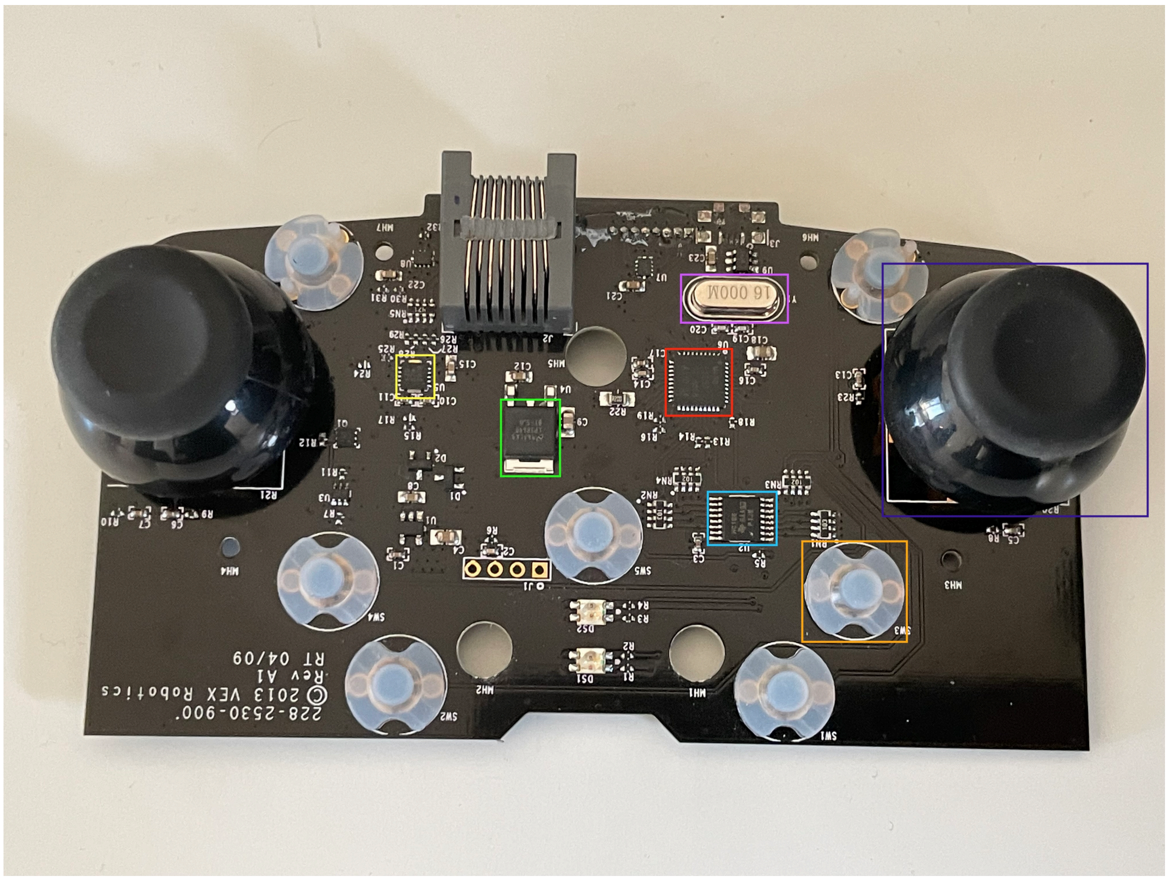
Team 13765C; London, United Kingdom

By Ali Juma

Why I chose to reverse engineer a controller

I have chosen to reverse engineer a VEX IQ Controller (1st Generation). But why would I want to reverse engineer a controller specifically? Firstly, a controller is ultimately a communication device. Thus, it intrigued me to understand how the VEX IQ Controller communicates. The controller has joysticks as well as buttons, so it has a variety of input methods that I want to understand. It is also powered by a battery, and can charge it by itself, so I am intrigued by the way it does that.

Components



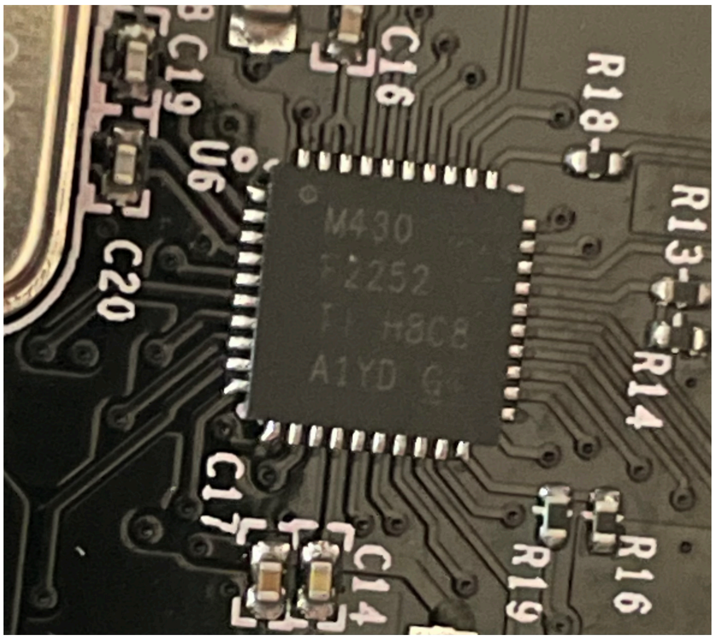
The front of the circuit board

After I opened it up, I found this, the circuit board. It has buttons, joysticks, and chips on it. I am going to identify the various components residing on it. I highlighted different components in various colours, and I will discuss them in turn.

Clock

This seems to be a quartz crystal clock, which are used to provide a steady pulse to synchronise the components. Reading the value printed on it, it seems to be a 16 MHz clock, which means that its signal oscillates around 16,777,216 times a second!

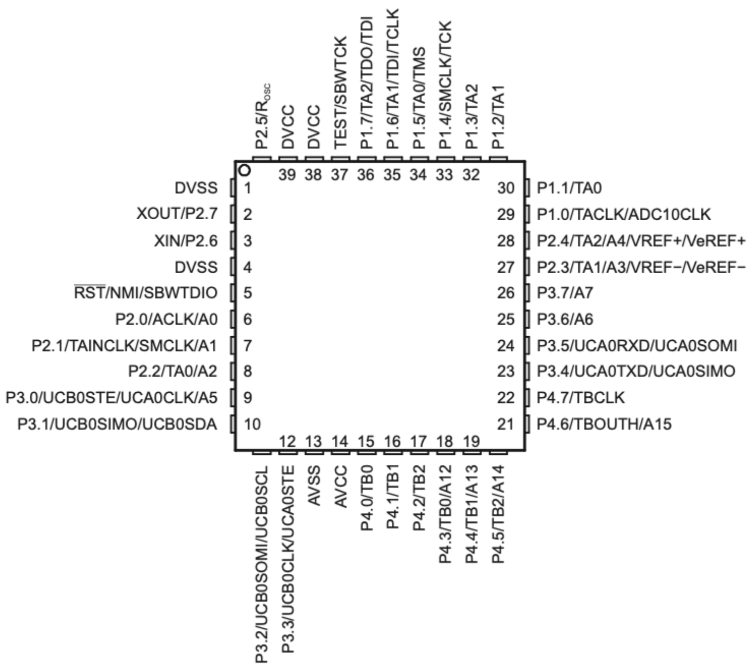
Microcontroller



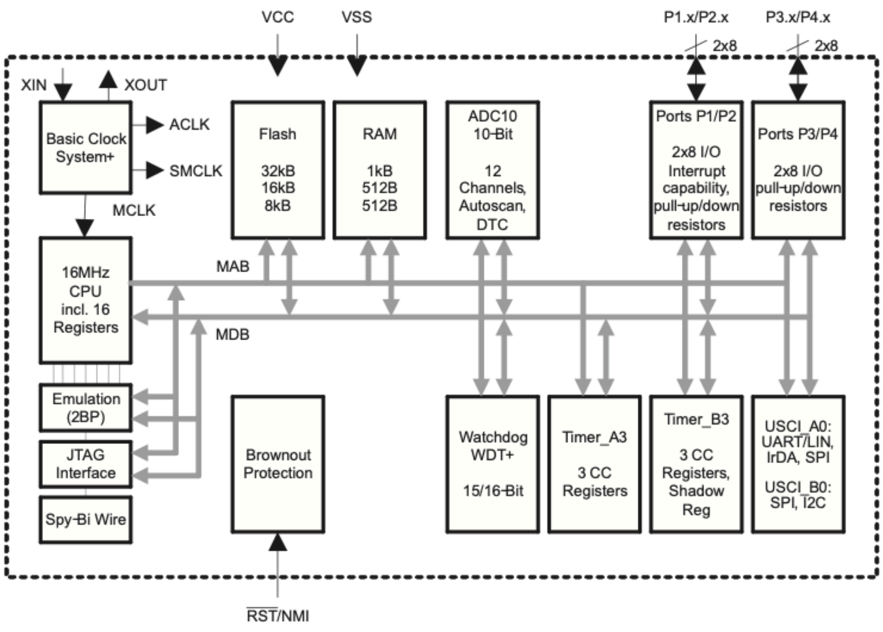
A closeup look at the microcontroller

This chip is the microcontroller, a small chip that drives the entire system. After some research, I found that this model is the Texas Instruments MSP430F2252, and it has 16 KB + 256B of flash memory and 512 bytes of RAM.

Looking at the data sheet, this is the pinout:

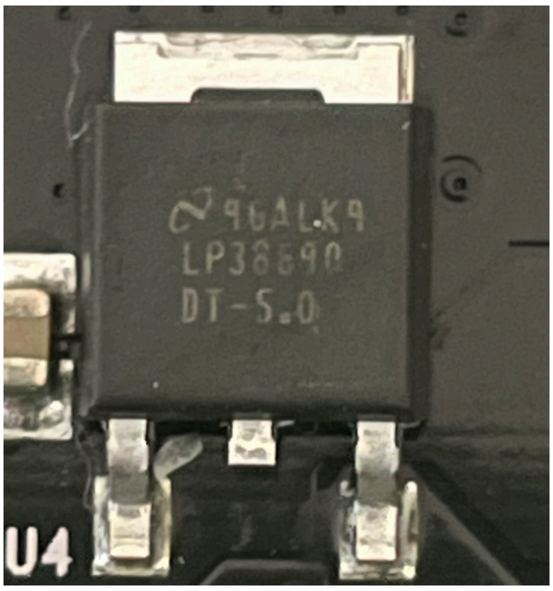


And this is a diagram of the inside:



It has many I/O ports and a small processor inside of it. It controls all the other various components in the controller.

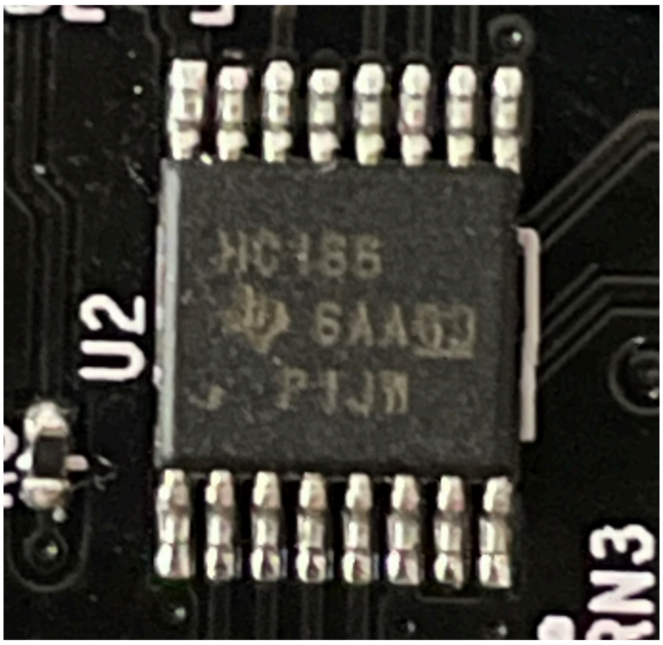
Linear regulator



A closeup look at the linear regulator

This chip is the LP38690 linear regulator, and it is used to manage the power going into a component. I found in the data sheet that it takes in anywhere from 2.7V to 10V, and outputs 1.8V. I am not sure what components it provides power for.

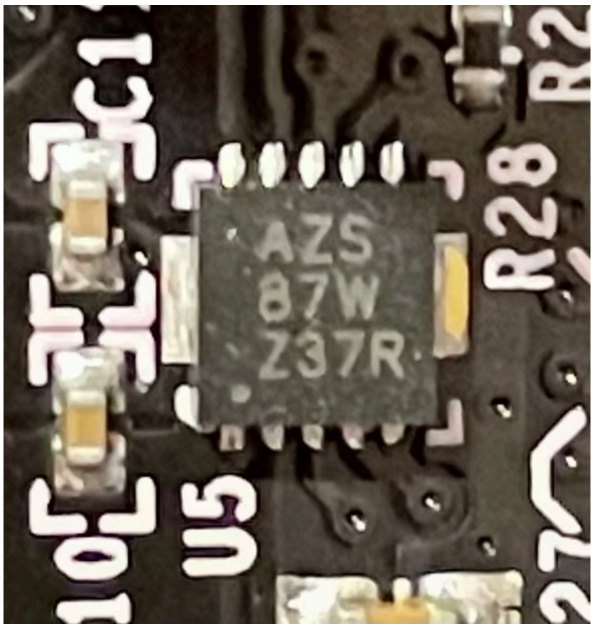
Shift register



A closeup look at the shift register

This chip is the HC166 shift register. Shift registers are used to store data, and output that data by shifting out bits one at a time. This one can hold 8 bits. Using a multimeter, I found that all the buttons and joysticks are connected to it, and that it is connected to the microcontroller, so I think that it holds the buttons' and joysticks' values and outputs them to the microcontroller.

Power management



A closeup look at the power management chip

This chip is the BQ24020, and it manages power. It accepts power over USB and from the battery and can tell the status of the battery, manages the charging of the battery, and outputs power for the other components.

Buttons

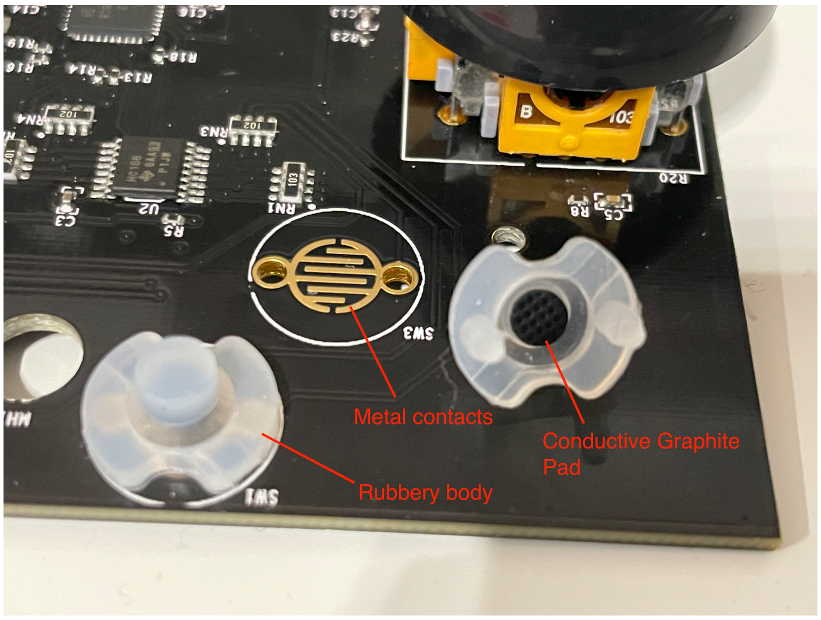


Diagram of buttons

The button mechanism is a rubbery body with a graphite pad inside of it, which is suspended over some metal contacts. When you press down on a button, the conductive graphite pad touches the metal contacts and connects them, which allows electricity to flow from one contact to another. This is then read by other components.

Joysticks

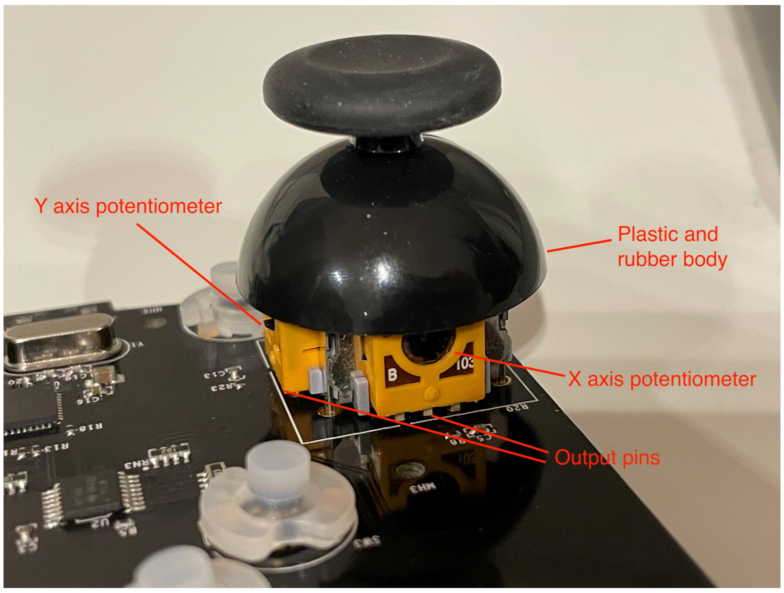
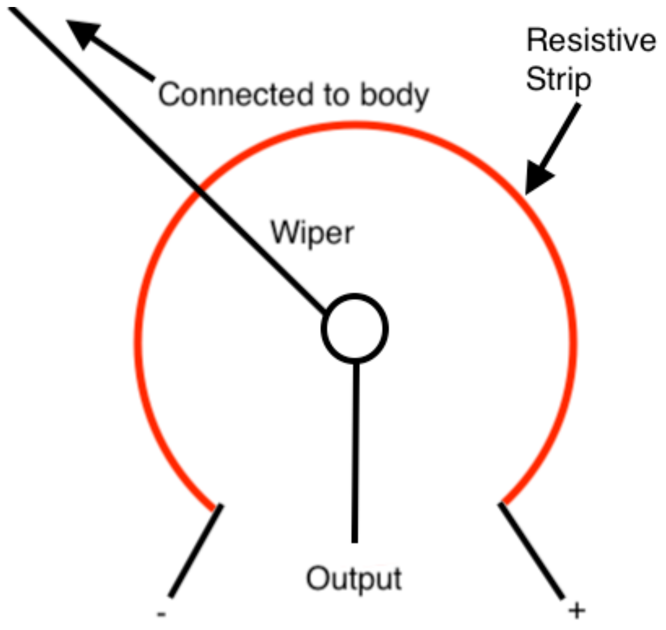


Diagram of one of the joysticks

A joystick is a plastic and rubber body that can rotate freely, connected to two potentiometers, one for each 2D axis. This is a simple diagram I made of one of the potentiometers, to illustrate how they work:



As you move a joystick along an axis, that axis' potentiometer will have a wiper that moves along a resistive strip. The resistive strip is connected between positive and negative terminals. The resistance is then put out to the output pin. The resistance of the output shows how far along that axis the joystick is. This is then read by components.

Conclusion

In conclusion, the VEX IQ controller is made up of a lot of parts, and they all work in an orchestrated way to communicate what the user is doing. I learnt how power from the battery and coming externally is managed by the device. In addition, I learnt how buttons and joysticks work, and how the signals from them are processed. I also attempted to connect a cable to the RJ45 port on the controller, which it uses to communicate, and tried to analyse the signals on the other end of the cable to figure out how it sends signals, but unfortunately I do not have an oscilloscope and I found nothing using a multimeter, so I did not put this in my report.