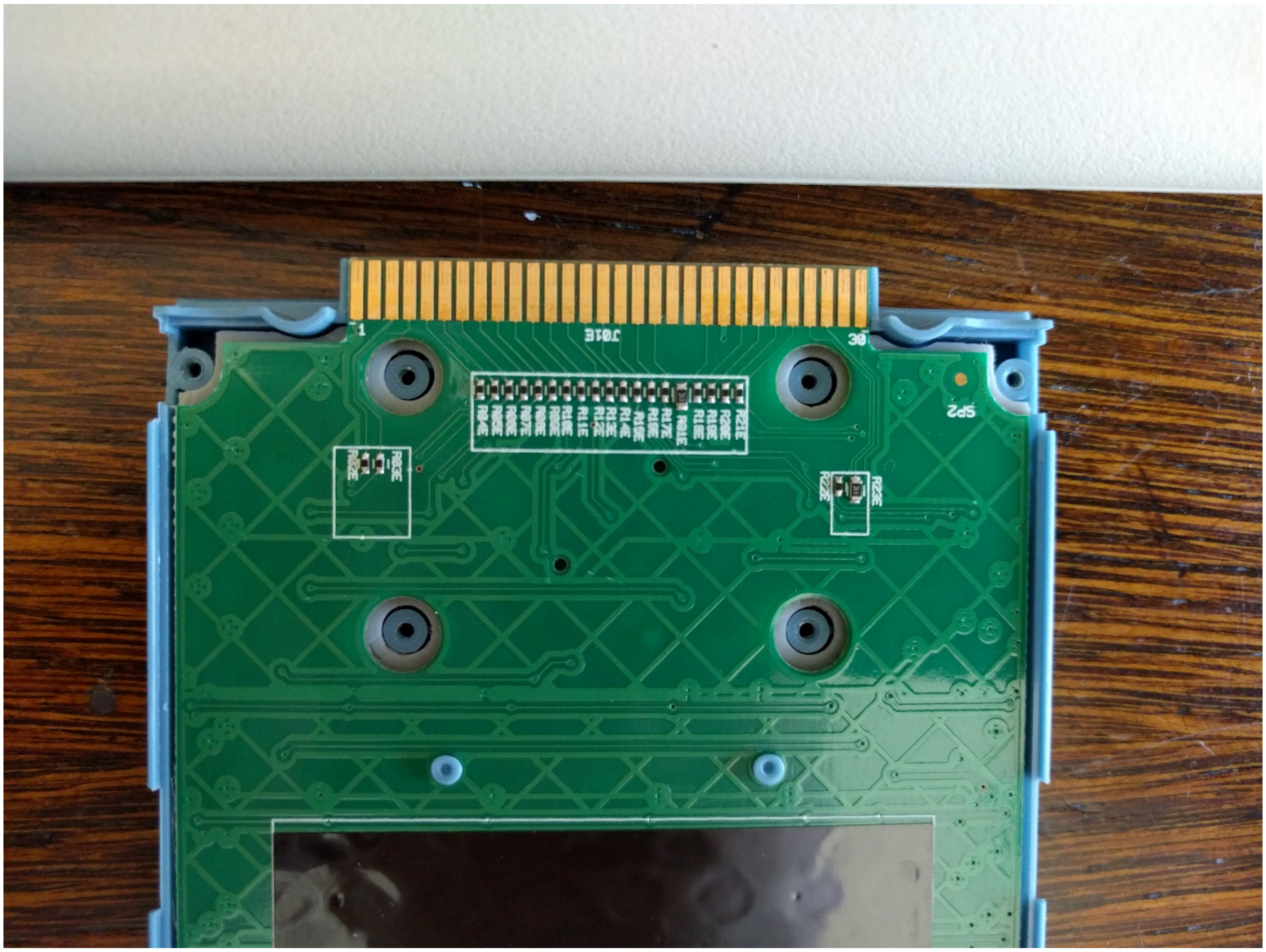
Interfacing with the TI-Nspire Removable Keypad Brain Dump

This calculator has a removable keypad with these pins on the back. This board is super simple to work with and uses a simple matrix circuit.

A very good article on how a matrix function is here: http://pcbheaven.com/wikipages/How_Key_Matrices_Works/

Getting it working

The keypad has 83 keys, and using a matrix layout means that 18 pins are the minimum amount required. (11pins x 8pins = 88 max keys) However, the keypad has 30 pins. So it is a fair assumption that many pins are left unused. Different keypads will have different pin requirements so it makes sense that this would happen. Before I connect it up I quickly took it apart to see if any of the pins are visibly unused.



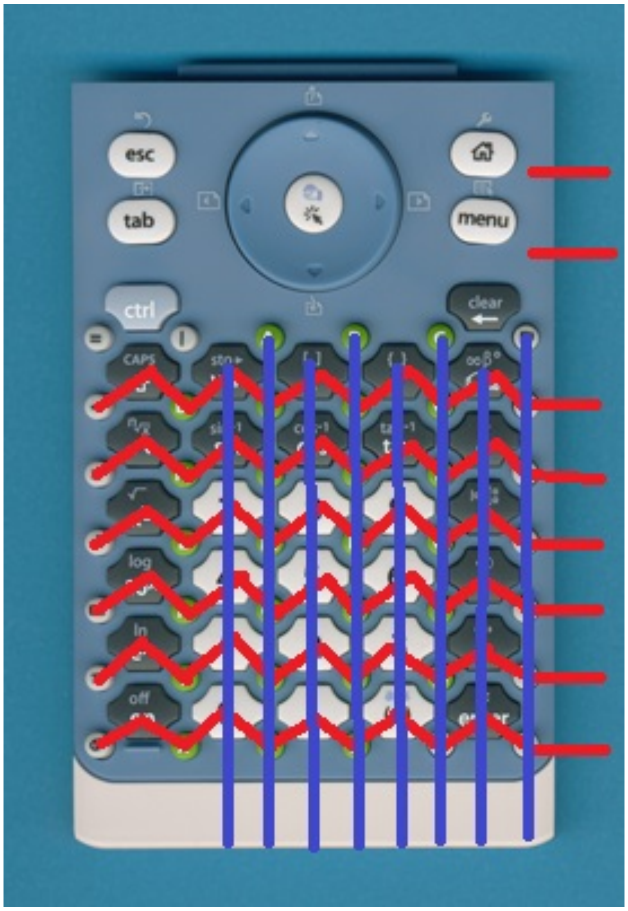
The pins are labeled 1-30. It looks like most of the pins have traces. I count 24 traces. By my earlier calculations we should have about 18 IO pins, and as per the website above 1 ground, and one VCC pin. So we have 4 pins that I am not sure the function of. This made me wonder if they are using a simple matrix or making it overly complex somehow. I never found the use of the 4 pins that have traces, but I can assume they proved to be functionally useless for my needs.

On the above website it mentions that the pins alternate column, row, column, row. With pins 1 being ground, and 2 being VCC. I wrote a really, really bad python program to detect the buttons:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

MATRIX = [  ["1A","1B","1C","1D","1E","1F","1G","1H","1I","1J","1k"],
        ["2A","2B","2C","2D","2E","2","2G","2H","2I","2J","2k"],
        ["3A","3B","3C","3D","3E","3F","3G","3H","3I","3J","3k"],
        ["4A","4B","4C","4D","4E","4F","4G","4H","4I","4J","4k"],
        ["5A","5B","5C","5D","5E","5F","5G","5H","5I","5J","5k"],
        ["6A","6B","6C","6D","6E","6F","6G","6H","6I","6J","6k"],
        ["7A","7B","7C","7D","7E","7F","7G","7H","7I","7J","7k"],
        ["8A","8B","8C","8D","8E","8F","8G","8H","8I","8J","8k"] ]

ROWRANGE = 8
COLRANGE = 11

ROW = [17,27,23,9,8,7,12,19] #19 is row not col (one of the 3)
COL = [16,26,20,22,18,25,24,6,13,5,4] #16 is col not row

for j in range(COLRANGE):
    GPIO.setup(COL[j], GPIO.OUT)
    GPIO.output(COL[j], 1)

for i in range(ROWRANGE):
    GPIO.setup(ROW[i], GPIO.IN, pull_up_down = GPIO.PUD_UP)

try:
    while(True):
        for j in range(COLRANGE):
            GPIO.output(COL[j],0)

            for i in range(ROWRANGE):
                if GPIO.input(ROW[i]) == 0:
                    print MATRIX[i][j]
                    time.sleep(0.2)
                    while(GPIO.input(ROW[i]) == 0):
                        pass

            GPIO.output(COL[j],1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

The code works by scanning/applying voltage to each row, while checking each column for a change in voltage. If a column goes high while a row has power going through it than we have the coordinates for the button pressed. I tried using wire clips to test the keyboard, but the pins were a little too small for that. I ended up soldering 30 awg wire to each pin because I could not find a pin header that had the correct widths.

The col, row, col, row pattern randomly stops after the 8 cols/rows. We are looking for the pinout of an 11*8 matrix.



So it would make sense to have it be col, row, col, row, row, row… right? After some trial and error the last 3 rows are on pins 23, 25, and 26.

Linux Driver

To be added

83 Pin Keypad Pinout

COL = column pin for matrix
ROW = row pin for matrix
DC = don’t know/care pin

1	GND
2	VCC
3	COL
4	ROW
5	COL
6	ROW
7	COL
8	ROW
9	COL
10	ROW
11	COL
12	ROW
13	COL
14	ROW
15	COL
16	ROW
17	COL
18	ROW
19	DC
20	DC
21	DC
22	DC
23	ROW
24	DC
25	ROW
26	ROW
27	DC
28	DC
29	DC
30	DC