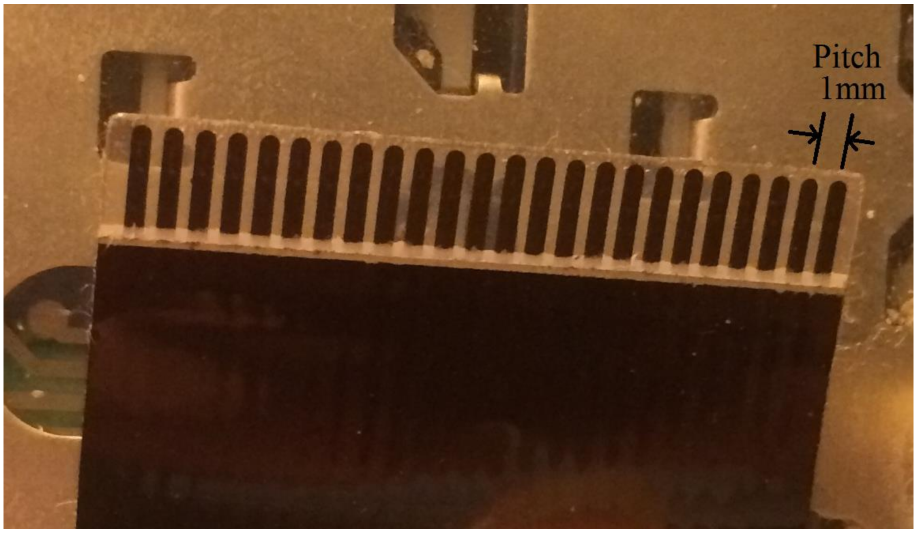
Instructable - Laptop Keyboard Conversion to USB

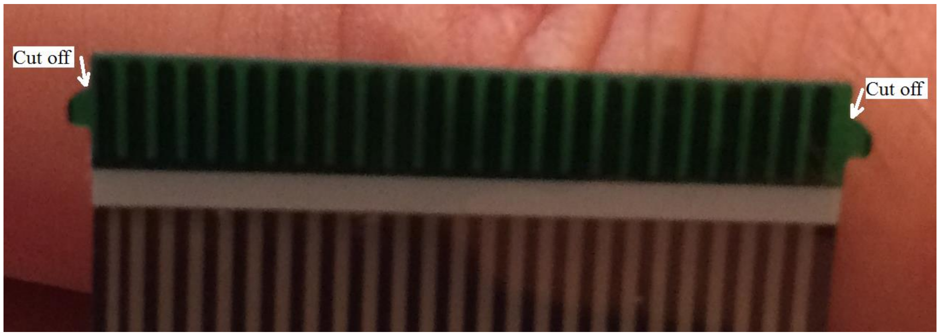
This Instructable will describe how to make a USB keyboard from your old laptop. It will use a Teensy microcontroller to scan the key switches. Teensies are often used by the mechanical keyboard enthusiasts at Geekhack and Deskthority and the TMK routine is the most popular software to decode the keys and output over USB. The alternative is to write your own program using the Teensyduino keyboard functions. Both methods need to know how your laptop key switches are wired. One approach, (that I never want to do again) is to exhaustively check every connector pin combination with an ohm meter while holding down a key. I did this when I converted a Sony Vaio into a Raspberry Pi laptop. An Instructable from alpinedelta disassembles the keyboard so the connections to each switch can be visually traced back to the connector. Instead of taking the keyboard apart or using an ohm meter, this Instructable will load the Teensy with an automated continuity tester. It will report over USB, the two pin numbers that are connected when you press a key. The program checks for continuity in both directions so a keyboard with diodes can still be tested. After every key has been pressed, the results can be transferred to a row-column matrix and used by the TMK keyboard controller software or a home-brew Teensyduino routine.

Laptop keyboards use a flexible printed circuit (FPC) that connects all the key switches in an array of rows and columns. There are many different FPC to motherboard connection methods but a large number of laptops use a set of traces that end with exposed metal on one side and a plastic backing on the other side. In order to fit in the connector, the plastic backing plus the FPC material will measure about 33 mm thick. A common FPC cable for a keyboard without a number pad has 24 or 25 signal traces with a 1 mm pitch. If there is a number pad, then it's common to have 26 traces with a 1 mm pitch. Most of the FPC cables I tested will fit in a generic connector with no modifications. Generic connectors for a keyboard FPC cable are readily available from Aliexpress. The number of signal traces and the pitch are the parameters you will need when ordering.

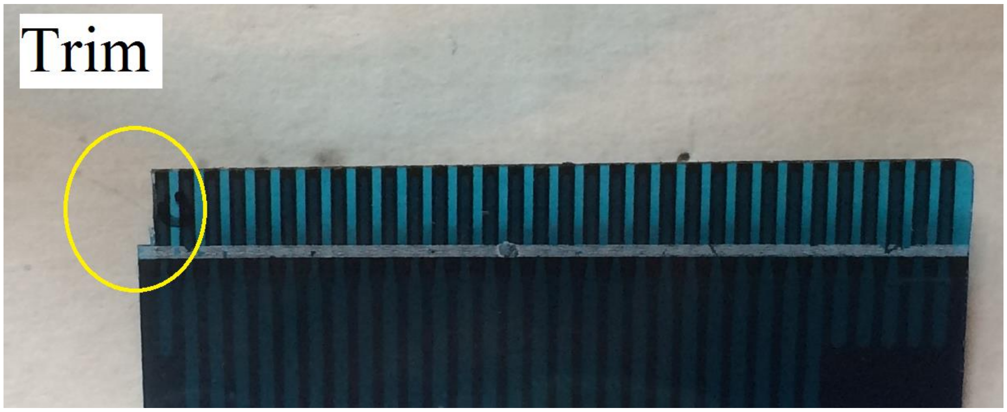
The picture below shows a standard 24 pin FPC Cable with a 1mm trace pitch.



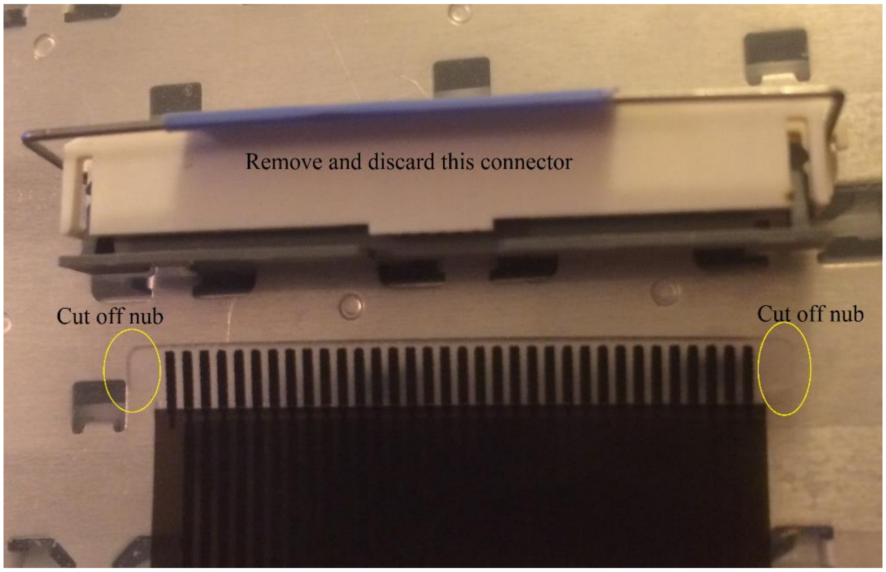
Some FPC cables need to be modified to fit in a generic connector. Locking nubs on the side of the cable are easy to remove with wire cutters.



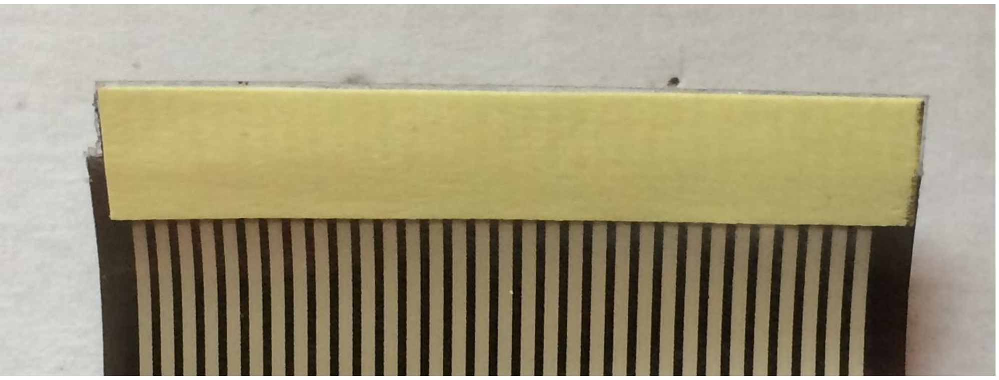
If the FPC traces don't line up with the connector pins, use an X-ACTO knife to trim along the side of the cable.



The Dell Latitude D630 keyboard needed the most modifications. It had a solder-less connector on the end of the FPC cable that was easily removed. Then I trimmed off the side nubs and made a notch to align the contacts.

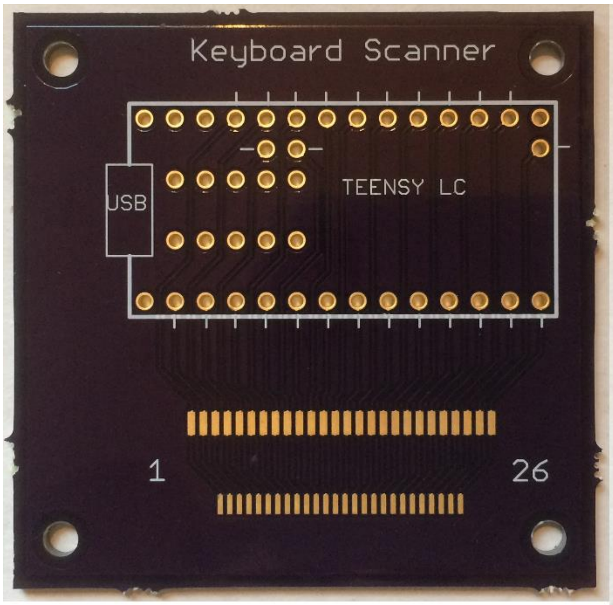


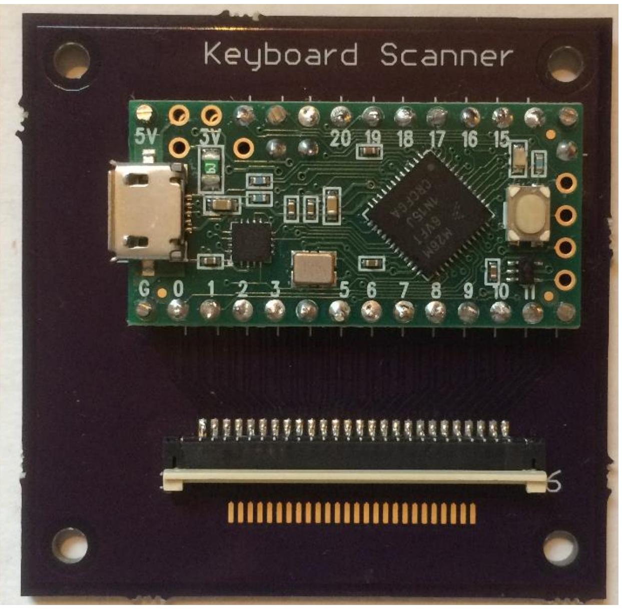
The final mod was to remove the extra thick plastic backing on the end of the cable and replace it with 2 pieces of a paper to bring the thickness back to normal for a generic FPC connector.



I designed a circuit board using Eagle for the Teensy LC that routes its 26 I/O pins to 26 surface mount pads for an FPC connector with a 1mm pitch. Just in case there is a keyboard out there with a smaller pitch, I also wired 26 pads with a 0.8mm pitch. A 24, 25, or 26 pin FPC connector can be soldered to this board based on your needs. I avoided using the 27th Teensy LC output because it’s connected to an LED and 27 pin FPC connectors are rare.

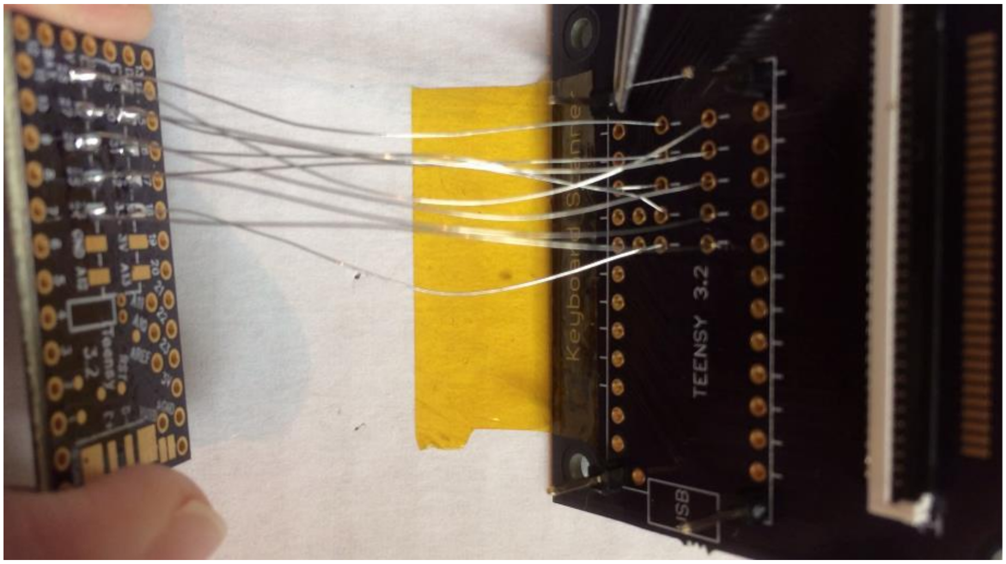
After soldering the FPC connector to the board, I soldered 4 header posts to the board to support the corners of the Teensy and then I soldered the Teensy to the header posts. The last step was to connect the rest of the Teensy I/O signals to the board with 30 gauge solid wire. I used wire instead of header posts to make it easy to cut the Teensy off the board for future projects. The Teensy pads that must be connected to the board are marked with a small line. The bare and assembled LC board is shown below.



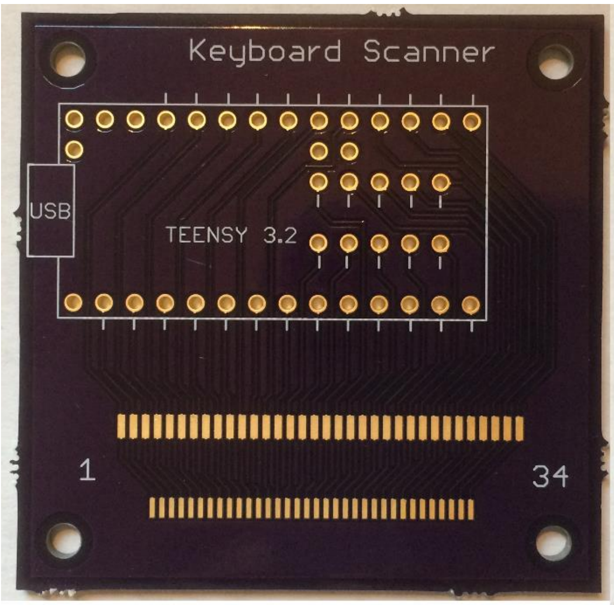


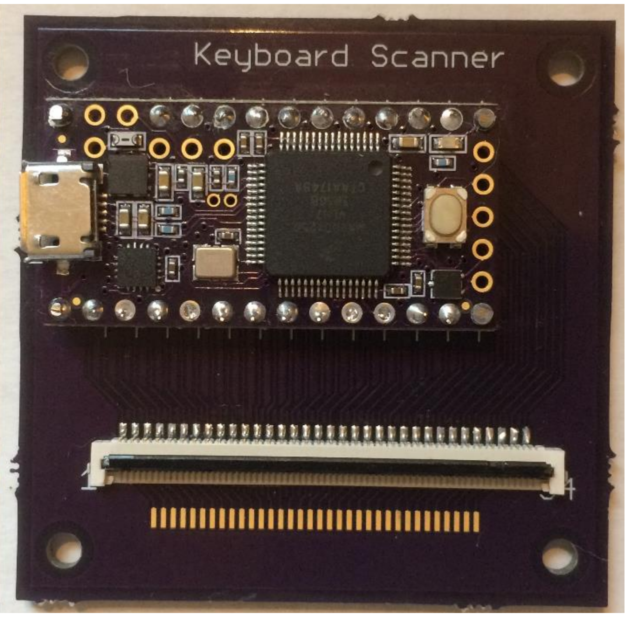
All of the signals for the Teensy LC are routed on one side of my board so I designed the other side for a Teensy 3.2 with 34 I/O signals and an FPC connector with a 1 mm or 0.8 mm pitch. If you need all 34 pins, you must unsolder the LED on the Teensy 3.2 to free it up for use by the keyboard.

The Teensy 3.2 uses surface mount pads for 10 of the I/O signals so it's a little more work to solder them to the board. After soldering the FPC connector to the 3.2 side of the board, solder 10 "flying leads" to the surface mount pads on the Teensy 3.2 and pass each wire thru the corresponding pad on the board.



Finish the assembly by soldering 30 gauge solid wire to the remaining I/O signals marked with a small line. The bare and assembled 3.2 board is shown below.





The Parts List is given below.

1. The Teensy LC is $15.53 or the Teensy 3.2 is $23.49 from Amazon. You can also order direct from PJRC.

2. The FPC connectors from AliExpress are about $5 for a lot of 5. An example search on their website would be “laptop keyboard connector 1.0 spacing 24 pin”. Digikey is another possible source but I haven't checked them out.

3. Fabrication of the circuit board costs $18 from OSH Park for a lot of 3, or $14 from DirtyPCBs for a lot of about 10. OSH Park fabricates the boards in the United States and takes about 12 days for delivery to Tacoma. DirtyPCBs are fabricated in China and take about a month to deliver to Tacoma. The EagleCad board file is located at my GitHub repo. If you don’t need to make any modifications, then you can send the Eagle board file directly to either vendor.

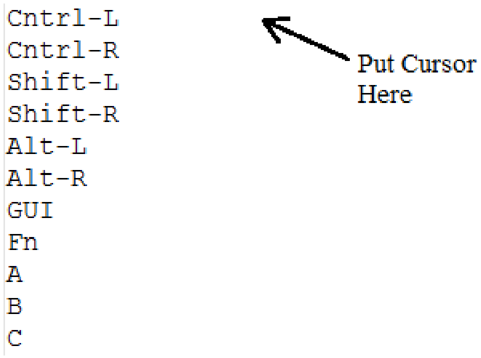
Other Items needed include header pins, wire, solder, flux, and USB cable.

Once you have the Teensy and FPC connector soldered to the board, follow these steps to decode your keyboard matrix.

Load the Continuity Tester Into the Teensy per the following procedure.

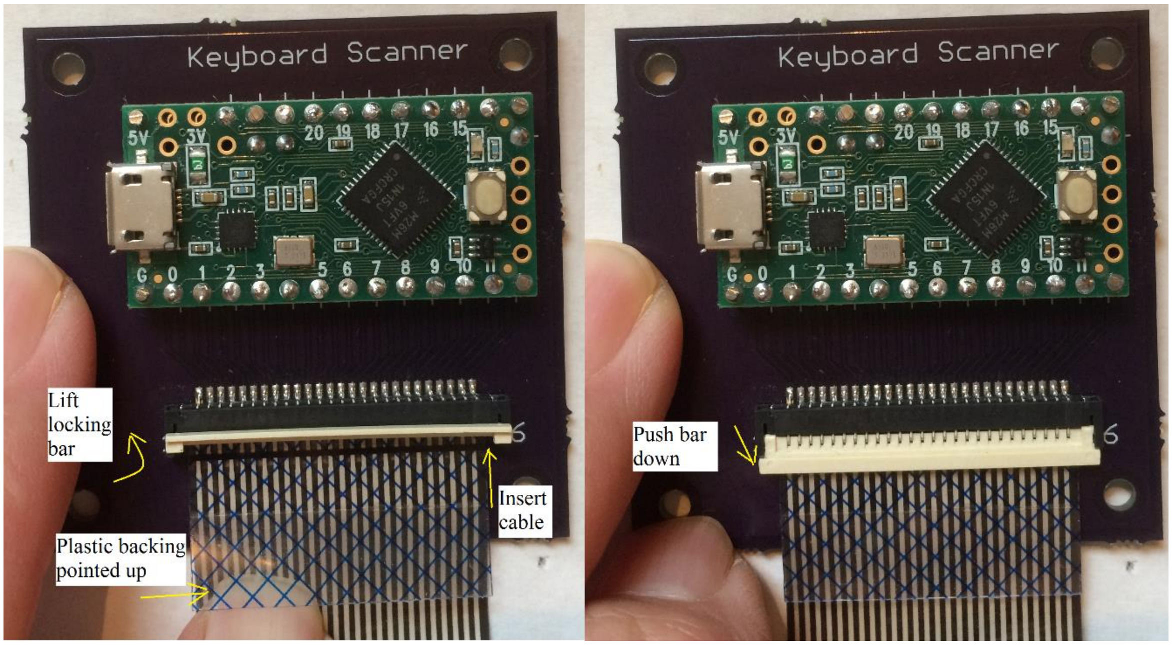
- Follow the PJRC link for installing Arduino and Teensyduino on your computer.
- Download the Arduino code from the Pin_Continuity_Tester folder at my GitHub repo. Use file Matrix_Decoder_LC.ino for a Teensy LC or Matrix_Decoder_3p2.ino for a Teensy 3.2.
- Load the Matrix_Decoder code into the Arduino integrated development environment (IDE).
- Connect a USB cable from the Teensy to the computer. Your computer should automatically load the necessary USB drivers.
- In the Arduino IDE, under “tools”, select board: Teensy LC or Teensy 3.2/3.1 depending on what you’re using. Also under “tools”, select USB type: Keyboard.
- Compile and load the Matrix_Decoder code into the Teensy. If it’s your very first time loading the Teensy, you’ll have to push the button on the Teensy to enable the loader.
- Disconnect the USB cable from the Teensy.

Open a text editor on your computer. I like to use Notepad++ on Windows or Geany on the Pi because they have column editing. There are two “key-list” text files you can download at my repo, one for a keyboard with a number pad and one for no number pad. The “key-list” file lists each key that you will push followed by tabs to make the results more readable and easy to copy into a spreadsheet. You will probably need to modify the file slightly to match your keyboard’s keys. A non-US keyboard can still use this routine, just make a list of your keys and the program will report pin connections. The GUI key in my list is either the "windows key" from a PC or the "clover key" from a Mac. Place the cursor to the right of the very first key in the list as shown below. This will determine where the Teensy begins to display the pin numbers as you push each key.

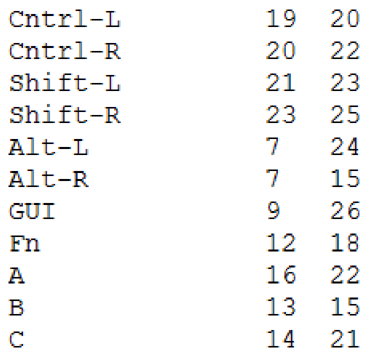


```
Cntrl-L
Cntrl-R
Shift-L
Shift-R
Alt-L
Alt-R
GUI
Fn
A
B
C
```

Use your finger nail to gently lift the locking bar on the FPC board connector to the open position. Slide the FPC cable into the connector with the bare metal contacts pointed down (closest to the board) and the plastic backing strip pointed up. Lock the cable to the connector by gently pushing the bar down. Connect a USB cable from the Teensy to the computer and wait 20 seconds for the Teensy to be recognized as a USB keyboard. This delay is in the code to make sure your computer is ready to receive numbers from the Teensy. If numbers are reported on the screen before any keys are pressed, these pins are shorted together and must be fixed. If you have an FPC cable with more than 26 signals, it may use some of the extra traces for grounds, back-lighting, or a track-pad. This may cause the test routine to register two pins as shorted. If this happens, you'll need to do some code modifications to exclude these pins. The code normally scans all pins starting from pin 1 and ending at the last pin but you can adjust these variables to avoid shorts.



Press each key, one by one on the test keyboard as listed on the editor screen. The Teensy will send two pin numbers over USB that were connected when the key was pressed. The Teensy will then send a down arrow to position the cursor for the next key. If your keyboard contains diodes on the switches, the first pin is the cathode side and the second pin is the anode side.



```
Cntrl-L         19  20
Cntrl-R         20  22
Shift-L         21  23
Shift-R         23  25
Alt-L           7   24
Alt-R           7   15
GUI             9   26
Fn              12  18
A               16  22
B               13  15
C               14  21
```

After pressing every key on the keyboard, save the finished file for analysis. To determine the Input and output pins, follow a few simple rules based on the Modifier keys; Control, Alt, Shift, GUI, and Fn. As a general rule, 8 of the keyboard pins will be inputs to the Teensy and the remainder will be outputs. The Modifier keys usually have an output row all to themselves which allows these keys to be held down while other keys are pressed. This avoids a sneak path which would cause ghosting. These "rules" are not always followed (especially by the Fn key) so you may need to do some trial and error as you build the matrix. I have lots of keyboard examples at my Github repo to help you out.

Control-Left and Control-Right will have a common pin between them. Example:

Cntrl-L	19 20
Cntrl-R	20 22

The common pin, Pin 20 in this example, will be a Teensy output, and 19 & 22 will be inputs.

Similarly Alt-Left and Alt-Right will have a common pin between them, just as Shift-Left and Shift-Right will also have a common pin. Example:

Alt-L	7 24
Alt-R	7 15
Shift-L	21 23
Shift-R	23 25

The Alt common pin will be a Teensy output, and 15 & 24 will be inputs.

The Shift common pin will be a Teensy output, and 21 & 25 will be inputs.

The GUI key is usually a single key as in this example;

GUI	9 26

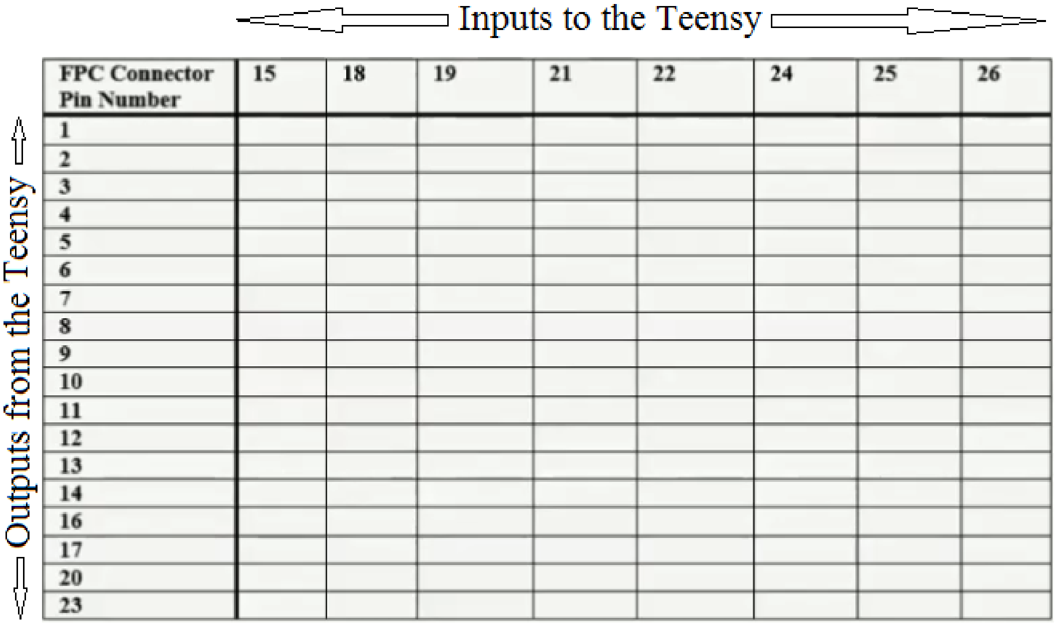
Search all the other pins in the list to see if 9 or 26 are used on other keys. In this example, pin 9 was not used for any other key so it will be a Teensy output and 26 will be an input. Sometimes both pins are used for other keys but one of the pins is used for common keys like letters and numbers and the other pin is used for less-common keys like page-up. In this case the pin used for common keys will be a Teensy input and the other pin will be an output. Note that the GUI key will still work if you swap the pins so don't overthink it.

The Fn key is also a single key as in this example;

Fn 12 18

Using the same approach as the GUI key, search all the other pins to see if 12 or 18 are used on other keys. In this example, pin 12 was not used for any other key so it will be an output and 18 will be an input. If both pins are used on other keys, follow the same rules as the GUI example. Sometimes both pins are used by common keys which means you can pick either pin as an input and the other as an output.

The eight input pins for the HP DV9000 example keyboard have been identified as; 15, 18, 19, 21, 22, 24, 25, and 26. All other pins will be Teensy outputs. Make a keyboard matrix table like the one shown below with the 8 input pins across the top and all the other pins as outputs on the side.



Inputs to the Teensy

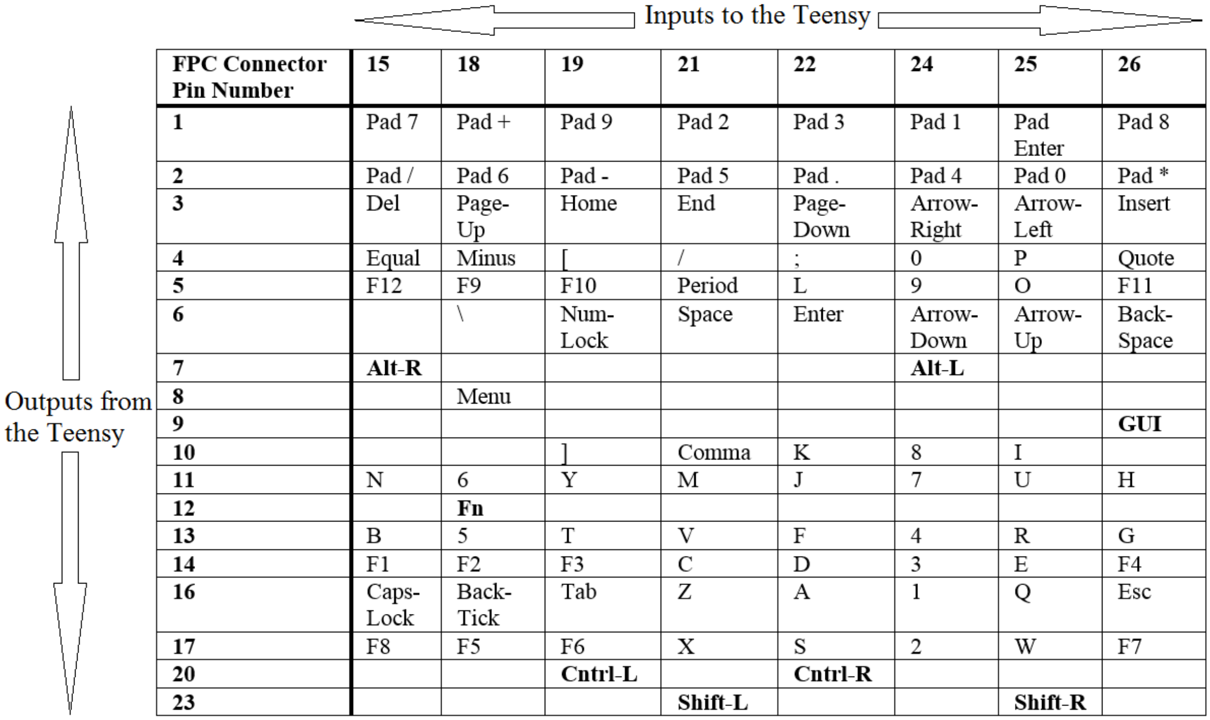
Outputs from the Teensy

FPC Connector Pin Number	15	18	19	21	22	24	25	26
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
16								
17								
20								
23								

The orientation of the keyboard matrix is just my personal preference and you can swap the rows/columns and inputs/outputs. Swapping pins may be necessary if you have a rare laptop keyboard that has diodes on its switches.

Sometimes only 7 pins can be identified as inputs because two modifier keys share the same input pin. If this happens, you'll have to make an educated guess for the 8th input. For some keyboards, the input and output pins are grouped together (i.e., 1 thru 8, and 9 thru 24) which makes it easy to fill in the missing input pin. Other keyboards have no grouping of inputs and outputs so you'll have to begin filling out the matrix with only 7 inputs. The missing input pin will reveal itself when some of the keys can't be placed in the matrix.

Place each key name at the row/column intersection of the pins as shown in the HP DV9000 keyboard example shown below. The modifier keys are in bold so you can see that they have a row all to themselves. This keyboard followed the “rules” exactly.



Inputs to the Teensy

Outputs from the Teensy

FPC Connector Pin Number	**15**	**18**	**19**	**21**	**22**	**24**	**25**	**26**
1	Pad 7	Pad +	Pad 9	Pad 2	Pad 3	Pad 1	Pad Enter	Pad 8
2	Pad /	Pad 6	Pad -	Pad 5	Pad .	Pad 4	Pad 0	Pad *
3	Del	Page-Up	Home	End	Page-Down	Arrow-Right	Arrow-Left	Insert
4	Equal	Minus	[	/	;	0	P	Quote
5	F12	F9	F10	Period	L	9	O	F11
6		\	Num-Lock	Space	Enter	Arrow-Down	Arrow-Up	Back-Space
7	**Alt-R**					**Alt-L**		
8		Menu						
9								**GUI**
10			]	Comma	K	8	I	
11	N	6	Y	M	J	7	U	H
12		**Fn**						
13	B	5	T	V	F	4	R	G
14	F1	F2	F3	C	D	3	E	F4
16	Caps-Lock	Back-Tick	Tab	Z	A	1	Q	Esc
17	F8	F5	F6	X	S	2	W	F7
20			**Cntrl-L**		**Cntrl-R**			
23				**Shift-L**			**Shift-R**	

You will need to translate the row and column FPC pin numbers to the Teensy I/O numbers for your keyboard controller code. The two tables below show how the FPC connector pins are routed to the Teensy LC I/O’s and the Teensy 3.2 I/O’s.

Teensy LC

FPC Pin #	Teensy LC I/O #
1	23
2	0
3	22
4	1
5	24
6	2
7	21
8	3
9	25
10	4
11	20
12	5
13	19
14	6
15	18
16	7
17	17
18	8
19	16
20	9
21	15
22	10
23	14
24	11
25	26
26	12

Teensy 3.2

FPC Pin #	Teensy 3.2 I/O #
1	23
2	0
3	22
4	1
5	21
6	2
7	20
8	3
9	19
10	4
11	18
12	5
13	17
14	6
15	24
16	7
17	25
18	8
19	33
20	9
21	26
22	10
23	27
24	11
25	28
26	12
27	32
28	31
29	30
30	29
31	16
32	15
33	14
34	13

To use your finished matrix with the TMK keyboard routine, follow the step by step instructions from matt3o at his Deskthority post. Hasu's TMK "wall of code" can be overwhelming and may have more features than you need so I wrote a simple USB keyboard routine using the Teensyduino "Micro-Manager" functions. It works fine if you just want a basic keyboard and can give you a starting point if you want to write your own keyboard controller code. Instructions on how to modify the LC code or the 3.2 code for your keyboard are provided at my GitHub repo. Every keyboard listed below has a folder at my repo containing a pin connect list, key matrix, and a Teensyduino USB keyboard routine giving you lots of examples to follow.

- Dell Inspiron 1525 - Keyboard Part Number D9K01
- Dell Latitude 131L - Keyboard Part Number V-0511BIAS1-US
- Dell Latitude X1 - Keyboard Part Number 0M6607
- Dell Latitude D630 - Keyboard Part Number DP/N 0DR160
- HP Compaq Presario 2100 - Keyboard Part Number AEKT1TPU011
- HP Compaq Presario V4000 - Keyboard Part Number NSK-H3L01
- HP Pavilion DV9000 - Keyboard Part number AEAT5U00110
- Sony Vaio PCG-K25 - Keyboard Part Number KFRMBA151B
- Sony Vaio VPCCW - Keyboard Part Number 148754321
- Sony Vaio VPCEA - Keyboard part number A-1765-621-A
- Sony Vaio VPCEB4 – Keyboard part number A-1766-425-A

I hope this Instructable will help you find new uses for your old broken laptop. Once you have the keyboard working, you can use a video converter card to drive the LCD. You can even make the touchpad work if your Teensy has two I/O pins left. The TMK code has a touchpad to USB option that can be enabled during the build process. If you are using Teensyduino, I have modified the touchpad code from Playground Arduino so it outputs over USB. The code and documentation are at my Github repo.

Send me an email at thedalles77@gmail.com if you have any questions, comments, or corrections.

Good Luck

Frank Adams