ProCAT Flash and Plover Hax

Contents

1 Overview

I want to use a ProCAT flash with Plover. The ProCAT Flash is a stenography machine. It allows for faster typing, and less strain. Plover is a FOSS Stenography project.

2 Work Log

2.1 Not All ProCAT Flash's will work with Plover

Some of the plover docs mention that a ProCAT flash will work with a RJ11 to DB9 cable, and that the protocol can be set to Baron. My device must be an older firmware as there is no option to change the protocol, and in fact the RJ11 appears to be only for outbound calling... There is a modem option, and it allows you to call a number. It appears to be for actually calling a phone line.

Instead, there is an arduino sketch of someone who has previously tapped into the serial out of the IO Expander (essentially the board that reads the keys), and helpfully outlined the process. By cutting the motherboard out of the picture, and driving the IO board directly from a 5V Arduino, one can use the ProCAT without the need for its built in modem. Neat.

2.2 Backup Plan: Reading from a ProCAT Flash with an Arduino

There is a six pin ribbon cable that can be cut in half, or removed, and then the arduino connected to it. This is a minimally invasive hack. I can always put the cable back. Let's begin.

The ink ribbon can be easily removed to gain better access to the IO board. The Shift Registers are TI 74HC165 which means this board may be able to be repaired, if needed. There is a conformal coating on the board, both sides. Funny how laptops don't bother with conformal coatings for waterproofing.[1]

[1] 100+ years of electronics engineering advances, billions of dollars of R&D, yet we can't figure out how to keep a laptop safe from a cup of H2O.

Vcc is pin 16, and Gnd is 8. Tracing that back to the 6 pin cable to confirm the pinout of the arduino sketch is right,

2.2.1 Never Believe the Internet

The pinout of the sketch was wrong, for my flash. To be fair, my flash is not a Stentura 200, so that's why. My pinout is:

```
On ribbon cable going from left to right looking from above.

RIBBON CABLE:
1 SH/LD (shift)
2 GND
3 CLK
4 PWR
5 Serial Out
6 ??? Goes to Resistor
(INSERT PICTURE)

On my IO board, the pins are staggered, so there is 1,2,3,
 then another column of 4,5,6 (columns start at the top,
1 being a square pad). Confusing.

IO BOARD PADS
1 ??? Goes to Resistor
2 PWR
3 GND
4 Serial Out
5 CLK (goes underneath a resistor, then to all clocks)
6 SH/LD
(INSERT PICTURE)
```

These pins on the IO board are also soldered to pads (not holes) and fragile. Shit design. Tiny wires. I disconnected one just trying to follow the wires. Also they put some tape w/grease on the wires, and have them in a different order on the PCB from what the ribbon cable wiring is. Shit. Nothing I can't handle, however.

There are three rails going on the top (bottom?) of the IO board. These are CLOCK, SH/LD, and Power. GND is a copper flood on the top (bottom?) of the IO board. Make sure to double check the pins line up with what you are connecting after building this.

If you look closely on the circuit board for the IO expander, you might see labels for the Pins. I have G for gnd, + for Pwr, C for clk, etc..

2.3 Soldering Experience is a plus

I don't need to use the ribbon cable. Instead, let's take off the wires and use my own cable. I might make a PCB to fit inside the Flash... I'll need to edit the Arduino sketch too.

Not only the IO board, but I'll also need to attach wires to the metal chassis or frame somewhere inside. The Pins of the Steno short the IO expander to ground so I need to have the Arduino connected to chassis as well.

Very carefully, I used solder wick to remove all solder from the existig pins. Making sure to tin the tip before using wick each time. After removing all the solder, I pushed through the pins to remove the conformal coating from the other side. Then I passed some wires through the (fairly thick) PCB. Not too hard, but it helps to have experience here.

2.4 steno-arduino sketch is also not compatible with Flash

I made some edits to the Arduino sketch. One thing noteworthy to myself is the

```
 /*
             * All inputs are pulled up. Pressing a key shorts
the circuit to
             * ground.
             *
             * We invert the logic here to convert to more
 conventional positive
             * logic.
             */
            pressed          = !digitalRead(DATA_IN);
```

Haven't see the !variable logic before. Only in booleans, if(!true)...

The mapping of the pins on the ProCAT flash is different from the Stentura, so some deciphering was necessary. I ended up brute forcing some of it, as I couldn't figure out exactly how it was different, with relation to the previous setup.

I also added the debounce logic from one of the repos into the code. And added some serial debug notes. All said and done, the device connected into Plover and worked without issue. Nice.

Here is the mapping that took 1-2 hours to decipher, and the end result (see also arduino folder).

```
void construct_data(char raw_data[], char packed_data[])
{
    packed_data[0] = 0x80;
    packed_data[1] = (raw_data[ 5] << 6)  /* S- */ //Have to move
 arrays entries around here, if you are going to
                   | (raw_data[ 4] << 4)  /* T- */ //change values.
 Need all 24 bits.
                   | (raw_data[ 3] << 3)  /* K- */
                   | (raw_data[ 2] << 2)  /* P- */
                   | (raw_data[ 1] << 1)  /* W- */
                   | (raw_data[ 0] << 0); /* H- */

    packed_data[2] = (raw_data[ 15] << 6)  /* R- */
                   | (raw_data[ 14] << 5)  /* A- */
                   | (raw_data[ 13] << 4)  /* O- */
                   | (raw_data[ 12] << 3); /* *  */

    packed_data[3] = (raw_data[11] << 3)  /* -E */
                   | (raw_data[10] << 2)  /* -U */
                   | (raw_data[9] << 1)  /* -F */
                   | (raw_data[8] << 0); /* -R */

    packed_data[4] = (raw_data[23] << 6)  /* -P */ // was 14
                   | (raw_data[22] << 5)  /* -B */  // was dupe 0
                   | (raw_data[21] << 4)  /* -L */
                   | (raw_data[20] << 3)  /* -G */
                   | (raw_data[19] << 2)  /* -T */
                   | (raw_data[18] << 1)  /* -S */
                   | (raw_data[17] << 0); /* -D */

    packed_data[5] = (raw_data[6] << 6)  /* #  */ //
                   | (raw_data[16] << 0); /* -Z */
}
```

The Gemini PR protocol is reasonable to deal with, and this hack worked out well. Looking at the above packed_data, you can see that all the Gemini PR protocol is, is the initial 0x80 byte, followed by the bits either being 0 or 1 for each key. Simple.

2.4.1 Simplest Explanation of a Shift In Register

I've used these 4 or 5 times now, but finally I understand a simple enough way to use these.

The **simplest shift register in** code does the following:

```
Hold All pins low (CLK,DATA,LATCH)

Bring Latch Up

    Enter For Loop

    Read DATA bit (into array)

    Increment clock up, then down (as long as Micro GPIO is <30MHz)
                                  (30MHz being max speed of shift ic)
    Read DATA bit

    Increment clock up, then down

    Repeat until for loop exhausted

Put Latch Down (now the data will again be read
                 by the parallel inputs)

Repeat
```

That is the simplest explanation for a shift in register, and it is demonstrated here, in the steno-arduino code. Nice.

3 Conclusion

The ProCAT Flash is similar to the Stentura 200 but not identical. Due to that, the arduino sketch needed to be customized somewhat to interface with the Shift Registers. Also, the pinout of the ribbon cable was different. Otherwise, thanks to the original efforts of the first steno-arduino sketch, I was able to make this old ProCAT Flash output to Plover without using the internal Motherboard, nor the RJ11 connection. I now have a real steno machine that cost no more than $100 including shipping. [2]

[2]This price is comparable to what it would've cost for a home built keyboard solution. But is the value the same? I doubt it. The machining of the key levers on this ProCAT

I feel using a genuine Steno machine was the right choice to starting down this road, as I don't want to deal with the toys or half baked solutions made by some hackers who are selling home built keyboards.[3] However, I will say, that once I'm comfortable with this, I might get one of those keyboards, add a few extra keys on[4], and make my own custom keyboard for interfacing with GNU Linux. There is a reason to have both a genuine steno machine, and a custom hacker diy keyboard solution. Why not both?

is miles beyond a PCB and some switches put together in a weekend.

[3]There is also the issue that the switches on these keyboard all require 10-30 something of force for each key, whereas the ProCAT only requires 10-20 of force for the first key, whereas all subsequent keys, are free of strain. A small difference, but the devil is in the details. You can't beat the real thing.

[4]By keys, I mean I plan to make a custom keyboard solution. Rotaries, Slide pots, etc... All the fun interfacing things.