

17:06 The DIP Flip Whixr Trick: An Integrated Circuit That Functions in Either Orientation

by Joe “Kingpin” Grand

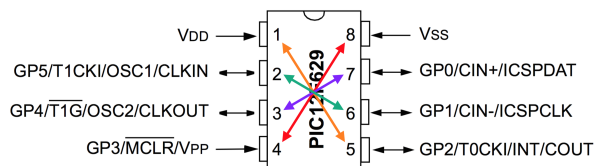
Hardware trickery comes in many shapes and sizes: implanting add-on hardware into a finished product, exfiltrating data through optical, thermal, or electromagnetic means, injecting malicious code into firmware, BIOS, or microcode, or embedding Trojans into physical silicon. Hackers, governments, and academics have been playing in this wide open field for quite some time and there’s no sign of things slowing down.

This PoC, inspired by my friend Whixr of #tymkrs, demonstrates the feasibility of an IC behaving differently depending on which way it’s connected into the system. Common convention states that ICs must be inserted in their specified orientation, assisted by the notch or key on the device identifying pin 1, in order to function properly.

So, let’s defy this convention!

Most standard chips, like digital logic devices and microcontrollers, place the power and ground connections at corners diagonal from each other. If one were to physically rotate the IC by 180 degrees, power from the board would connect to the ground pin of the chip or vice versa. This would typically result in damage to the chip, releasing the magic smoke that it needs to function. The key to this PoC was finding an IC with a more favorable pin configuration.

While searching through microcontroller data sheets, I came across the Microchip PIC12F629. This particular 8-pin device has power and GPIO (General Purpose I/O) pins in locations that would allow the chip to be rotated with minimal risk. Of course, this PoC could be applied to any chip with a suitable pin configuration.



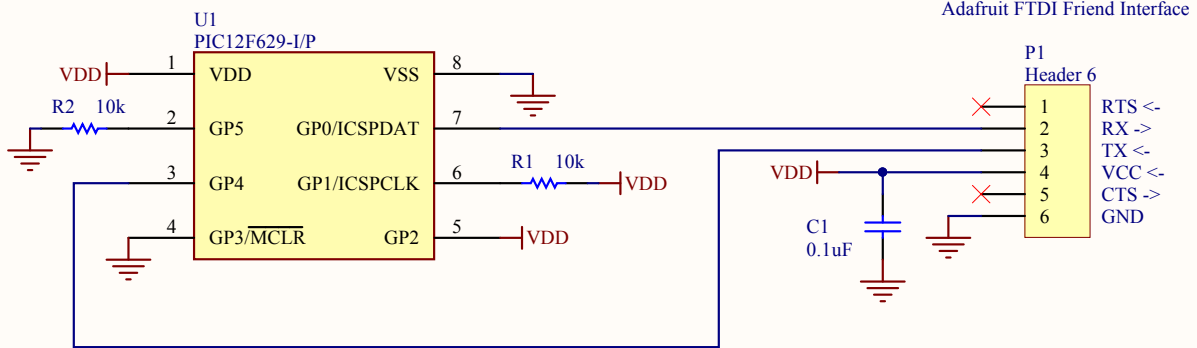
In the pinout drawing, which shows the chip from above in its normal orientation, arrows denote the alternate functionality of that particular pin when the chip is rotated around. Since power (VDD) is normally connected to pin 1 and ground (VSS) is normally connected to pin 8, if the chip is rotated, GP2 (pin 5) and GP3 (pin 4) would connect to power and ground instead. By setting both GP2 and GP3 to inputs in firmware and connecting them to power and ground, respectively, on the board, the PIC will be properly powered regardless of orientation.

I thought it would be fun to change the data that the PIC sends to a host PC depending on its orientation.

On power-up of the PIC, GP1 is used to detect the orientation of the device and set the mode accordingly. If GP1 is high (caused by the pull-up resistor to VCC), the PIC will execute the normal code. If GP1 is low (caused by the pull-down resistor to VSS), the PIC will know that it has been rotated and will execute the alternate code. This orientation detection could also be done using GP5, but with inverted polarity.

The PIC’s UART (asynchronous serial) output is bit-banged in firmware, so I’m able to reconfigure the GPIO pins used for TX and RX (GP0 and GP4) on-the-fly. The TX and RX pins connect directly to an Adafruit FTDI Friend, which is a standard FTDI FT232R-based USB-to-serial adapter. The FTDI Friend also provides 5V (VDD) to the PoC.

In normal operation, the device will look for a key press on GP4 from the FTDI Friend’s TX pin and then repeatedly transmit the character ‘A’ at 9600 baud via GP0 to the FTDI Friend’s RX pin. When the device is rotated 180 degrees, the device will look for a key press on GP0 and repeatedly transmit the character ‘B’ on GP4. As a key press detector, instead of reading a full character from the host, the device just looks for a high-to-low transition on the PIC’s currently configured RX pin. Since that pin idles high, the start bit of any data sent from the FTDI Friend will be logic low.



```

switch (input(PIN_A1)) { // orientation
  detection
2  case MODE_NORMAL: // normal behavior
  #use rs232(baud=9600, bits=8, parity=N,
  stop=1, xmit=PIN_A0, force_sw)
4
  //wait for a keypress
6  while(input(PIN_A4));

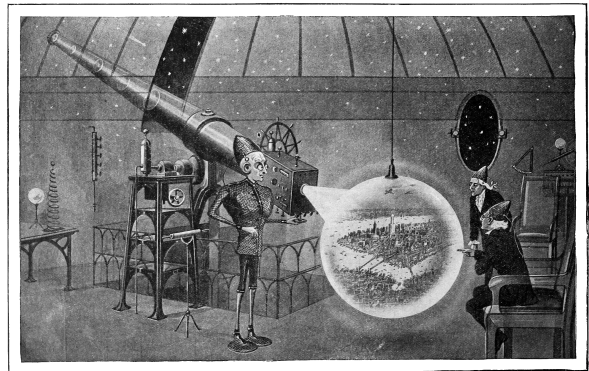
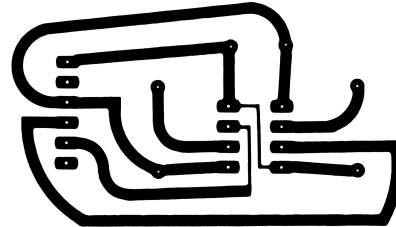
8  while(1){
  printf("A ");
10  delay_ms(10);
  }
12  break;

14 case MODE_ALTERNATE: // abnormal behavior
  #use rs232(baud=9600, bits=8, parity=N,
  stop=1, xmit=PIN_A4, force_sw)
16
  // wait for a keypress
18  while(input(PIN_A0));

20  while(1){
  printf("B ");
22  delay_ms(10);
  }
24  break;
}

```

Let this PoC serve as a reminder that one should not take anything at face value. There are an endless number of ways that hardware, and the electronic components within a hardware system, can misbehave. Hopefully, this little trick will inspire future hardware mischief and/or the development of other sneaky circuits. If nothing else, you're at least armed with a snarky response for the next time some over-confident engineer insists ICs will only work in one direction!



For your viewing entertainment, a demonstration of my breadboard prototype can be found on Youtube.¹⁷ Complete engineering documentation, including schematic, bill-of-materials, source code, and layout for a small circuit board module are also available.¹⁸

¹⁷Joe Grand, Sneaky Circuit: This DIP Goes Both Ways

¹⁸unzip pocorgtfo17.pdf dipflip.zip # or at www.grandideastudio.com/portfolio/sneaky-circuits/