PS/2 Keyboard Tutorial

Introduction

This tutorial provides an overview on how to use the PS/2 input on the DE1- SoC board.

> ⚠ USB-to-PS/2 converters do *NOT* seem to work for most USB keyboards. If you are going to use a keyboard, either find an old PS/2 keyboard, or an old USB keyboard that comes with its own converter.

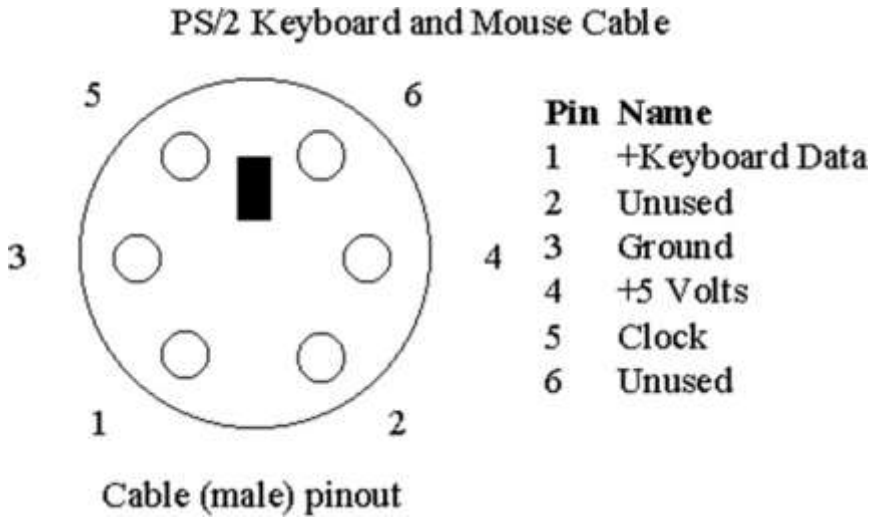

Figure 1: *Front view of a PS/2 cable pinout.*

The pinout for the PS/2 Keyboard and Mouse cable is shown above. Through a clock pin and a serial data pin, the FPGA connects to the clock and serial line keyboard data line of the PS/2 cable. The clock signal and serial keyboard data is handled by the provided Verilog files and should not be a concern of the user.

Data Representation

When a key is pressed, released, or held down, a packet of information known as a **scan code** is sent by the keyboard over the PS/2 data line. Each scan code is categorized as either a “make code” or “break code.” When a key is *pressed* or *held down*, a **make code** is sent. While a key is pressed down, the make code is continually sent at a certain interval. When a key is *released*, a **break code** is sent. Every key has a unique make code and break code.

There are three standard scan code sets, named one, two, and three. The default for modern keyboards is **set two**. The scan code sets are included in the zip for this set of drivers and documentation as `.pdf` files with tables for the scan codes.

Behavior of Multiple Keys Held

If one key is held down, its make code is sent at a periodic rate. If another key is then held down, the make code of the new key held down is sent at a periodic rate instead. If the first key is released, the break code for the first key is sent once and the make code of the second key continues to stream at a periodic rate. If the second key is released, the break code for the second key is sent and no new scan codes are sent until a new key is pressed or the first key is released.

The number of keys that can be held down following this behavior depends on the circuitry of the keyboard used. It can be assumed that 2 keys can be held down for any keyboard and for any 2 keys. If too many keys are held down, then the scan code 0x00 is sent at a periodic rate until the keys over the maximum are released.

Interface for Keyboard

The provided interface only works with keyboards that use scan code set 2 or 3, which happens to encompass most modern keyboards.

`keyboard_press_driver.v` describes a module that returns keyboard data when a key is either pressed or released.

When a key is pressed, a `makeBreak` bit is set to show where a make or break was sent. Recall that makes are key presses and breaks are key releases. The `outCode` byte register is set with the keyboard data. This byte is the scan code for the given key. Note that a break code is usually 2-3 bytes, but only the unique last byte is given because it is enough to identify the key and it is almost always the same as the one byte make code.

- User Inputs
  - `CLOCK_50` – 50 MHz clock
  - `reset` – active high reset signal
- Driver Outputs
  - `valid` – 1 if outputs are valid data and 0 otherwise
  - `makeBreak` – 1 for make and 0 for break
  - `outCode` – a byte (`[7:0]`) representing keyboard data (take note of this one-byte representation from the description above)
- Top-Level Signals
  - PS2_DAT – ps/2 data line
  - PS2_CLK – ps/2 clock

Users should connect the inputs and outputs to their design and the top-level inputs should connect to the top-level pins of the same name.

Tutorial and code developed by Kyle Gagner and Jesse Liston with Professor Scott Hauck
`keyboard_inner_driver.v` and `oneshot.v` are adapter from/provided by Dr. John S. Loomis, Professor Emeritus of Electrical and Computer Engineering at the University of Dayton. http://www.johnloomis.org/