The IBM PC Keyboard

Original keyboard connector: DIN-5

The PS/2 Mini-DIN 6 Connector

Like RS-232, but with a clock.
Odd parity, one start, one stop.
Keyboard-to-host shown: keyboard initiates everything.

Communication to the Keyboard

Host-generated

Keyboard-generated clock

Host-brings Clock low, then Data low to indicate transfer to keyboard, then releases Clock (rises).
Keyboard starts generating clock signals. Host supplies serial data, changing after each falling edge. After stop bit, host releases Data. Keyboard pulls Data low for one more clock signal to indicate it received the byte.

Synchronous Serial Interface

Start LSB B1 B2 B3 B4 B5 B6 MSB Parity Stop

ED LED control
Caps lock Num lock Scroll lock

EE Echo

Keyboard will respond with EE

F0 Set scan code set
Keyboard will respond with FA and wait for another byte 01–03. 00 leaves scan code unchanged.

F3 Set key repeat rate
Keyboard responds with FA and waits for second byte, indicating repeat rate.

Codes (Keyboard to Host)

00/FF Error or buffer overflow
F0 Key-up
FA Acknowledge
EE Echo response
FE Resend
F0 Extended code coming

Commands (Host to Keyboard)

F4 Enable keyboard
Responds with FA, clears buffer, enables scanning.

F5 Disable keyboard
Responds with FA, disables keyboard.

FE Resend
Retransmit the last byte.

FF Reset Keyboard
PS/2 Mouse Protocol

Three bytes sent every time mouse moves or button clicked:

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>Middle</td>
<td>Right</td>
</tr>
</tbody>
</table>

Movement values are since last transmission: 9-bit two’s-complement (signed) numbers.

Many more variants, modes, and other junk.

Using the PS/2 Port w/ Digilent Bd.

Add lines in the .UCF file about PS2C and PS2D:

```c
NET "PS2D" LOC=m15;  # Data
NET "PS2C" LOC=m16;  # Clock
```

Add these ports in the "add cores" dialog. Make the clock pin a falling-edge-sensitive interrupt.

In the .MHS file, this appears as:

```c
PORT PS2D = ps2io_GPIO_in, DIR = IN;
PORT PS2C = PS2C, DIR = IN,
          SIGIS = INTERRUPT,
          SENSIVITY = EDGE_FALLING;
```

Using the PS/2 Port w/ Digilent Bd.

This makes a keyboard-to-host port. Interrupt signals a clock. Do the shift register in software.

Register the handler and enable interrupts:

```c
microblaze_enable_interrupts();
XIntc_RegisterHandler(
    XPAR_OPB_INTC_0_BASEADDR,
    XPAR_OPB_INTC_0_SYSTEM_PS2C_INTR,
    XPAR_PS2IO_BASEADDR,
    (void*)ps2_int_handler,
    XPAR_PS2IO_BASEADDR,
    XPAR_RS232_INTERRUPT_MASK | XPAR_SYSTEM_PS2C_MASK);
XIntc_mMasterEnable(XPAR_OPB_INTC_0_BASEADDR);
XIntc_mEnableIntr(
    XPAR_OPB_INTC_0_BASEADDR,
    XPAR_RS232_INTERRUPT_MASK | XPAR_SYSTEM_PS2C_MASK);
```

Interrupt Handler

```c
#define SIZE 16
unsigned char buffer[SIZE];
int head = 0; int tail = 0;
unsigned int code = 0; unsigned int bit = 11;
void ps2_int_handler(void* baseaddr_p) {
    int next;
    code = (code >> 1) |
         (XGpio_mReadReg(XPAR_PS2IO_BASEADDR, XPAR_GPIO_DATA_OFFSET) << 9);
    if (--bit == 0) {
        next = (head + 1) & (SIZE - 1);
        if (next != tail) {
            buffer[head] = code;
            head = next;
        }
        bit = 11; code = 0;
    }
    XGpio_mWriteReg( XPAR_PS2IO_BASEADDR, XPAR_GPIO_ISR_OFFSET, 1);
}
```

Buffer Management Routines

```c
int character_available()
    {  
        if (result;
            microblaze_disable_interrupts();
            result = (head = tail);
            microblaze_enable_interrupts();
            return result;
    }

    unsigned char get_character()
    {  
        unsigned char result;
            microblaze_disable_interrupts();
            result = buffer[tail];
            tail = (tail + 1) & (SIZE - 1);
            microblaze_enable_interrupts();
            return result;
    }```