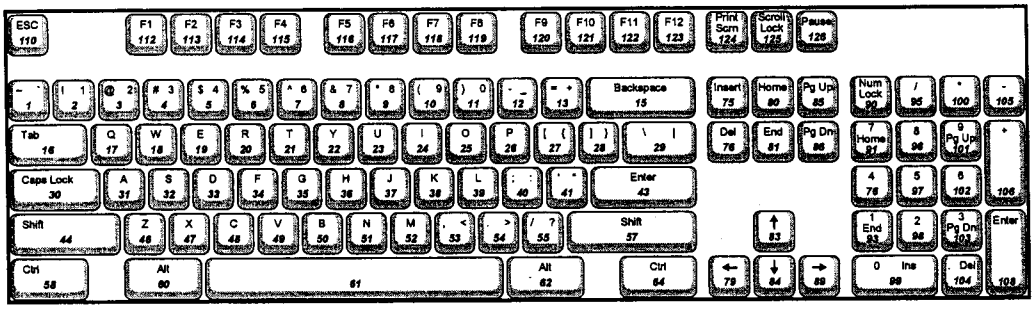
Communications: Interfacing to the PS/2 Keyboard



10 Communications: Interfacing to the PS/2 Keyboard

The Altera UP 1 and UP 1X boards support the use of either a mouse or keyboard using a PS/2 connector on the board. This provides only the basic electrical connections from the PS/2 cable and the FLEX chip. It is necessary to design a hardware interface using logic in the FLEX chip to communicate with a keyboard or a mouse. Serial-to-parallel conversion using a shift register is required.

10.1 PS/2 Port Connections

The PS/2 port consists of 6 pins including ground, power (VDD), keyboard data, and a keyboard clock line. The UP 1 board supplies the power to the mouse or keyboard. Two lines are not used. The keyboard data line is pin 31 on the FLEX chip, and the keyboard clock line is pin 30. Pins must be specified in one of the design files. Both the clock and data lines are open collector and bi-directional. The clock line is normally controlled by the keyboard, but it can also be driven by the computer system or in this case the FLEX chip, when it wants to stop data transmissions from the keyboard. Both the keyboard and the system can drive the data line. The data line is the sole source for the data transfer between the computer and keyboard. The keyboard and the system can exchange several commands and messages as seen in Tables 10.1 and 10.2.

Table 10.1 PS/2 Keyboard Commands and Messages.

Commands Sent to Keyboard	Hex Value
Reset Keyboard Keyboard returns AA, 00 after self-test	FF
Resend Message	FE
Set key typematic (autorepeat) XX is scan code for key	FB, XX
Set key make and break	FC, XX
Set key make	FD, XX
Set all key typematic, make and break	FA
Set all keys make	F9
Set all keys make and break	F8
Make all keys typematic (autorepeat)	F7
Set to Default Values	F6
Clear Buffers and start scanning keys	F4
Set typematic (autorepeat) rate and delay Set typematic (autorepeat) rate and delay Bits 6 and 5 are delay (250ms to 1 sec) Bits 4 to 0 are rate (all 0's-30x/sec to all 1's 2x/sec)	F3, XX
Read keyboard ID Keyboard sends FA, 83, AB	F2
Set scan code set XX is 01, 02, or 03	F0, XX
Echo	EE
Set Keyboard LEDs XX is 00000 Scroll, Num, and Caps Lock bits 1 is LED on and 0 is LED off	ED, XX

Table 10.2 PS/2 Commands and messages sent by keyboard.

Messages Sent by Keyboard	Hex Value
Resend Message	FE
Two bad messages in a row	FC
Keyboard Acknowledge Command Sent by Keyboard after each command byte	FA
Response to Echo command	EE
Keyboard passed self-test	AA
Keyboard buffer overflow	00

10.2 Keyboard Scan Codes

Keyboards are normally encoded by placing the key switches in a matrix of rows and columns. All rows and columns are periodically checked by the keyboard encoder or "scanned" at a high rate to find any key state changes. Key data is passed serially to the computer from the keyboard using what is known as a scan code. Each keyboard key has a unique scan code based on the key switch matrix row and column address to identify the key pressed.

There are different varieties of scan codes available to use depending on the type of keyboard used. The PS/2 keyboard has two sets of scan codes. The default scan code set is used upon power on unless the computer system sends a command the keyboard to use an alternate set. The typical PC sends commands to the keyboard on power up and it uses an alternate scan code set. To interface the keyboard to the UP 1 board, it is simpler to use the default scan code set since no initialization commands are required.

10.3 Make and Break Codes

The keyboard scan codes consist of 'Make' and 'Break' codes. One make code is sent every time a key is pressed. When a key is released, a break code is sent. For most keys, the break code is a data stream of F0 followed by the make code for the key. Be aware that when typing, it is common to hit the next key(s) before releasing the first key hit.

Using this configuration, the system can tell whether or not the key has been pressed, and if more than one key is being held down, it can also distinguish which key has been released. One example of this is when a shift key is held down. While it is held down, the '3' key should return the value for the '#' symbol instead of the value for the '3' symbol. Also note that if a key is held down, the make code is continuously sent via the typematic rate until it is released, at which time the break code is sent.

10.4 The PS/2 Serial Data Transmission Protocol

The scan codes are sent serially using 11 bits on the bi-directional data line. When neither the keyboard nor the computer needs to send data, the data line and the clock line are High (inactive).

As seen in Figure 10.1, the transmission of a single key or command consists of the following components:

1. A start bit ('0')
2. 8 data bits containing the key scan code in low to high bit order
3. Odd parity bit such that the eight data bits plus the parity bit are an odd number of ones
4. A stop bit ('1')

The following sequence of events occur during a transmission of a command by the keyboard:

1. The keyboard checks to ensure that both the clock and keyboard lines are inactive. Inactive is indicated by a High state. If both are inactive, the keyboard prepares the 'start' bit by dropping the data line Low.
2. The keyboard then drops the clock line Low for approximately 35us.
3. The keyboard will then clock out the remaining 10 bits at an approximate rate of 70us per clock period. The keyboard drives both the data and clock line.
4. The computer is responsible for recognizing the 'start' bit and for receiving the serial data. The serial data, which is 8 bits, is followed by an odd parity bit and finally a High stop bit. If the keyboard wishes to send more data, it follows the 12th bit immediately with the next 'start' bit.

This pattern repeats until the keyboard is finished sending data at which point the clock and data lines will return to their inactive High state. In Figure 10.1 the keyboard is sending a scan code of 16 for the "1" key and it has a zero parity bit.

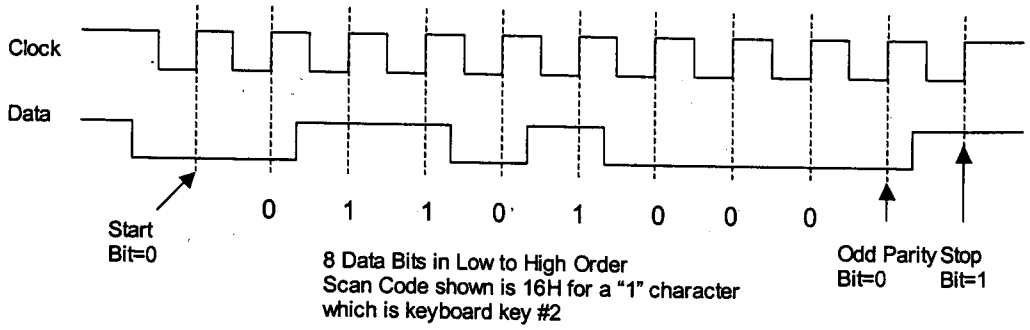


Figure 10.1 Keyboard Transmission of a Scan Code.

When implementing the interface code, it will be necessary to filter the slow keyboard clock to ensure reliable operation with the fast logic inside the FLEX chip. Whenever an electrical pulse is transmitted on a wire, electromagnetic properties of the wire cause the pulse to be distorted and some portions of the pulse may be reflected from the end of the wire. On some PS/2 keyboards and mice there is a reflected pulse on the cable that is strong enough to cause additional clocks to appear on the clock line.

Here is one approach that solves the reflected pulse problem. Feed the PS/2 clock signal into an 8-bit shift register that uses the 25Mhz clock. AND the bits of the shift register together and use the output of the AND gate as the new "filtered" clock. This prevents noise and ringing on the clock line from causing occasional extra clocks during the serial-to-parallel conversion in the FLEX chip.

A few keyboards and mice will work without the clock filter and many will not. They all will work with the clock filter, and it is relatively easy to implement. This circuit is included in the UP1cores for the keyboard and the mouse.

As seen in Figure 10.2, the computer system or FLEX chip sends commands to the PS/2 keyboard as follows:

1. System drives the clock line Low for approximately 60us to inhibit any new keyboard data transmissions. The clock line is bi-directional.
2. System drives the data line Low and then releases the clock line to signal that it has data for the keyboard.
3. The keyboard will generate clock signals in order to clock out the remaining serial bits in the command.
4. The system will send its 8-bit command followed by a parity bit and a stop bit.
5. After the stop bit is driven High, the data line is released.

Upon completion of each command byte, the keyboard will send an acknowledge (ACK) signal, FA, if it received the data successfully. If the system does not release the data line, the keyboard will continue to generate the clock, and upon completion, it will send a 're-send command' signal, FE or FC, to the system. A parity error or missing stop bit will also generate a re-send command signal.

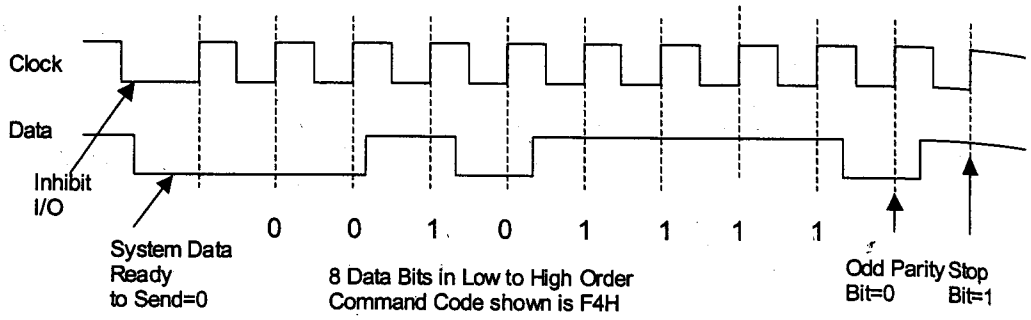


Figure 10.2 System Transmission of a Command to PS/2 Device.

10.5 Scan Code Set 2 for the PS/2 Keyboard

PS/2 keyboards are available in several languages with different characters printed on the keys. A two-step process is required to find the scan code. A key number is used to lookup the scan code. Key numbers needed for the scan code table are shown in Figure 10.3 for the English language keyboard layout.

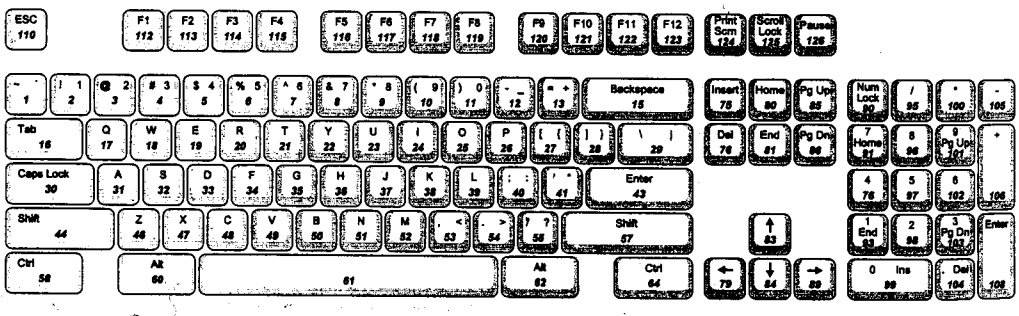


Figure 10.3 Key Numbers for Scan Code.

Each key sends out a make code when hit and a break code when released. When several keys are hit at the same time, several make codes will be sent before a break code.

The keyboard powers up using this scan code as the default. Commands must be sent to the keyboard to use other scan code sets. The PC sends out an initialization command that forces the keyboard to use the other scan code.

The interface is much simpler if the default scan code is used. If the default scan code is used, no commands will need to be sent to the keyboard. The keys in Table 10.3 for the default scan code are typematic (i.e. they automatically repeat the make code if held down).

Table 10.3 Scan Codes for PS/2 Keyboard.

Key#	Make Code	Break Code	Key#	Make Code	Break Code	Key#	Make Code	Break Code
1	0E	F0 0E	31	1C	F0 1C	90	77	F0 77
2	16	F0 16	32	1B	F0 1B	91	6C	F0 6C
3	1E	F0 1E	33	23	F0 23	92	6B	F0 6B
4	26	F0 26	34	2B	F0 2B	93	69	F0 69
5	25	F0 25	35	34	F0 34	96	75	F0 75
6	2E	F0 2E	36	33	F0 33	97	73	F0 73
7	36	F0 36	37	3B	F0 3B	98	72	F0 72
8	3D	F0 3D	38	42	F0 42	99	70	F0 70
9	3E	F0 3E	39	4B	F0 4B	100	7C	F0 7C
10	46	F0 46	40	4C	F0 4C	101	7D	F0 7D
11	45	F0 45	41	52	F0 52	102	74	F0 74
12	4E	F0 4E	43	5A	F0 5A	103	7A	F0 7A
13	55	F0 55	44	12	F0 12	104	71	F0 71
15	66	F0 66	46	1A	F0 1A	105	7B	F0 7B
16	0D	F0 0D	47	22	F0 22	106	79	F0 79
17	15	F0 15	48	21	F0 21	110	76	F0 76
18	1D	F0 1D	49	2A	F0 2A	112	05	F0 05
19	24	F0 24	50	32	F0 32	113	06	F0 06
20	2D	F0 2P	51	31	F0 31	114	04	F0 04
21	2C	F0 2C	52	3A	F0 3A	115	0c	F0 0C
22	35	F0 35	53	41	F0 41	116	03	F0 03
23	3C	F0 3C	54	49	F0 49	117	0B	F0 0B
24	43	F0 43	55	4A	F0 4A	118	83	F0 83
25	44	F0 44	57	59	F0 59	119	0A	F0 0A
26	4D	F0 4D	58	14	F0 14	120	01	F0 01
27	54	F0 54	60	11	F0 11	121	09	F0 09
28	5B	F0 5B	61	29	F0 29	122	78	F0 78
29	5D	F0 5D	62	E0 11	E0 F0 11	123	07	F0 07
The remaining key codes are a function of the shift, control, alt, or num-lock keys.								

Table 10.3 (Continued) - Scan Codes for PS/2 Keyboard.

Key	No Shift or Num Lock		Shift*		Num Lock On	
#	Make	Break	Make	Break	Make	Break
76	E0 70	E0 F0 70	E0 F0 12 E0 70	E0 F0 70 E0 12	E0 12 E0 70	E0 F0 70 E0 F0 12
76	E0 71	E0 F0 71	E0 F0 12 E0 71	E0 F0 71 E0 12	E0 12 E0 71	E0 F0 71 E0 T0 12
79	E0 6B	E0 F0 6B	E0 F0 12 E0 6B	E0 F0 6B E0 12	E0 12 E0 6B	E0 F0 6B E0 F0 12
80	E0 6C	E0 F0 6C	E0 F0 12 E0 6C	E0 F0 6C E0 12	E0 12 E0 6C	E0 F0 6C E0 F0 12
81	E0 69	E0 F0 69	E0 F0 12 E0 69	E0 F0 69 E0 12	E0 12 E0 69	E0 F0 69 E0 F0 12
83	E0 75	E0 F0 75	E0 F0 12 E0 75	E0 F0 75 E0 12	E0 12 E0 75	E0 F0 75 E0 F0 12
84	E0 72	E0 F0 72	E0 F0 12 E0 72	E0 F0 72 E0 12	E0 12 E0 72	E0 F0 72 E0 F0 12
85	E0 7D	E0 F0 7D	E0 F0 12 E0 7D	E0 F0 7D E0 12	E0 12 E0 7D	E0 F0 7D E0 F0 12
86	E0 7A	E0 F0 7A	E0 F0 12 E0 7A	E0 F0 7A E0 12	E0 12 E0 7A	E0 F0 7A E0 F0 12
89	E0 74	E0 F0 74	E0 F0 12 E0 74	E0 F0 74 E0 12	E0 12 E0 74	E0 F0 74 E0 F0 12

* When the left Shift Key is held down, the 12 - FO 12 shift make and break is sent with the other scan codes. When the right Shift Key is held down, 59 – FO 59 is sent.

Key	Scan Code		Shift Case *	
#	Make	Break	Make	Break
95	E0 4A	E0 F0 4A	E0 F0 12 E0 4A	E0 12 F0 4A

* When the left Shift Key is held down, the 12 - FO 12 shift make and break is sent with the other scan codes. When the right Shift Key is held down, 59 - FO 59 is sent. When both Shift Keys are down, both sets of codes are sent with the other scan codes.

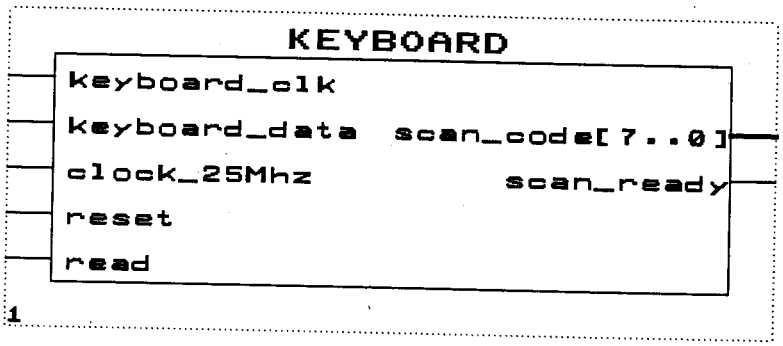
Key	Scan Code		Control Case, Shift Case		Alt Case	
#	Make	Break	Make	Break	Make	Break
124	E0 12 E0 7C	E0 F0 7C E0 F0 I2	E0 7C	E0 F0 7C	84	F0 84

Key #	Make Code	Control Key Pressed
126 *	El 14 77 El F0 14 F0 77	E0 7E E0 F0 7E

* This key does not repeat

10.6 The Keyboard UP1core

The following VHDL code for the keyboard UP1core shown in Figure 10.4 reads the scan code bytes from the keyboard. In this example code, no command is ever sent to the keyboard, so clock and data are always used as inputs and the keyboard power-on defaults are used. To send commands, a more complex bi-directional tri-state clock and data interface is required. The details of such an interface are explained in the next chapter on the PS/2 mouse. The keyboard powers up and sends the self-test code AA and 00 to the FLEX chip before it is downloaded.




```
LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.STD_LOGIC_ARITH.ALL;
USE  IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY keyboard IS
    PORT(  keyboard_clk, keyboard_data, clock_25Mhz ,
           reset, read          : IN    STD_LOGIC;
           scan_code            : OUT   STD_LOGIC_VECTOR( 7 DOWNTO 0 );
           scan_ready           : OUT   STD_LOGIC);
END keyboard;

ARCHITECTURE a OF keyboard IS
    SIGNAL INCNT                    : STD_LOGIC_VECTOR( 3 DOWNTO 0 );
    SIGNAL SHIFTIN                  : STD_LOGIC_VECTOR( 8 DOWNTO 0 );
    SIGNAL READ_CHAR                : STD_LOGIC;
    SIGNAL INFLAG, ready_set        : STD_LOGIC;
    SIGNAL keyboard_clk_filtered    : STD_LOGIC;
    SIGNAL filter                   : STD_LOGIC_VECTOR( 7 DOWNTO 0 );

BEGIN
    PROCESS ( read, ready_set )
    BEGIN
        IF read = '1' THEN
          scan_ready <= '0';
        ELSIF ready_set'EVENT AND ready_set = '1' THEN
          scan_ready <= '1';
        END IF;
    END PROCESS;
                    --This process filters the raw clock signal coming from the
                    -- keyboard using a shift register and two AND gates
Clock_filter:
    PROCESS
        BEGIN
```

```
        WAIT UNTIL clock_25Mhz'EVENT AND clock_25Mhz = '1';
        filter ( 6 DOWNTO 0 ) <= filter( 7 DOWNTO 1 ) ;
        filter( 7 ) <= keyboard_clk;
        IF filter = "11111111" THEN
          keyboard_clk_filtered <= '1';
        ELSIF filter = "00000000" THEN
          keyboard_clk_filtered <= '0';
        END IF;
    END PROCESS Clock_filter;
            --This process reads in serial scan code data coming from the keyboard
    PROCESS
    BEGIN
      WAIT UNTIL (KEYBOARD_CLK_filtered'EVENT AND KEYBOARD_CLK_filtered = '1');
      IF RESET = '1' THEN
          INCNT  <= "0000";
          READ_CHAR   <= '0';
      ELSE
          IF KEYBOARD_DATA = '0' AND READ_CHAR = '0' THEN
              READ_CHAR   <= '1';
              ready_set   <= '0';
          ELSE
                      -- Shift in next 8 data bits to assemble a scan code
              IF READ_CHAR = '1' THEN
                  IF INCNT < "1001" THEN
                      INCNT   <= INCNT + 1;
                      SHIFTIN( 7 DOWNTO 0 ) <= SHIFTIN( 8 DOWNTO 1 );
                      SHIFTIN( 8 )      <= KEYBOARD_DATA;
                      ready_set   <= '0';
                          -- End of scan code character, so set flags and exit loop
                  ELSE
                      scan_code  <= SHIFTIN( 7 DOWNTO 0 );
                      READ_CHAR   <='0';
                      ready_set   <= '1';
                      INCNT  <= "0000";
                  END IF;
              END IF;
          END IF;
    END IF;
END PROCESS;
END a;
```

The keyboard clock is filtered in the Clock_filter process using an 8-bit shift register and an AND gate to eliminate any reflected pulses, noise, or timing hazards that can be found on some keyboards. The clock signal in this process is the 25Mhz-system clock. The output signal, keyboard_clk_filtered, will only change if the input signal, keyboard_clk, has been High or Low for eight successive 25Mhz clocks or 320ns. This filters out noise and reflected pulses on the keyboard cable that could cause an extra or false clock signal on the fast FLEX chip. This problem has been observed to occur on some PS/2 keyboards and mice and is fixed by the filter routine.

The RECV_KBD process waits for a start bit, converts the next eight serial data bits to parallel, stores the input character in the signal, charin, and sets a flag, scan_ready, to indicate a new character was read. . The scan_ready or input ready flag is a handshake signal needed to ensure that a new scan code is read in and processed only once. Scan_ready is set whenever a new scan code is received. The input signal, read, resets the scan ready handshake signal.

The process using this code to read the key scan code would need to wait until the input ready flag, scan_ready, goes High. This process should then read in the new scan code value, scan_code. Last, read should be forced High and Low to clear the scan_ready handshake signal.

Since the set and reset conditions for scan_ready come from different processes each with different clocks, it is necessary to write a third process to generate the scan_ready handshake signal using the set and reset conditions from the other two processes. Hitting a common key will send a 1-byte make code and a 2-byte break code. This will produce at least three different scan_code values each time a key is hit and released.

A shift register is used with the filtered clock signals to perform the serial to parallel conversion. No command is ever sent the keyboard and it powers up using scan code set 2. Since commands are not sent to the keyboard, in this example clock and data lines are not bi-directional. The parity bit is not checked.

10.7 A Design Example Using the Keyboard UP1core

Here is a simple design using the Keyboard UP1core. The last byte of the scan code will appear in the seven-segment LED display. Read and Scan_ready are not used in this example.