

Reading Keyboard Scan Codes Through the PS/2 Interface on the XS board

TSTE70 TSEI20

ES

2004-3-24

Table of Contents

Chapter 1 Keyboard	3
1.1 Reading Keyboard Scan Codes Through the PS/2 Interface on the XS board	3
1.1.1 Background	3
1.1.2 Assignment	4
1.1.3 Problem definition	4
1.1.4 Interface	6
1.1.5 Requirements	6
1.2 Design	7
1.2.1 Introduction	7
1.2.2 Adding the pin attributes	8
1.2.3 Creating the testbench	9
1.3 Synthesis flow (using LeonardoSpectrum)	9
1.3.1 Synthesize the Design	9
1.3.2 Results	10
1.4 Downloading the Design	10
Appendix A PC AT keyboard ref	13
A.1 Description of keyboard	13
A.1.1 make code	13
A.1.2 break code	14
A.1.3 key code	14
A.1.4 scan code	14
A.1.5 Keyboard serial data	15
A.2 Scancodes and commands	15
A.2.1 Keyboard layouts with codes	15
A.2.2 Scan code tables	16
A.3 Further reading	17
A.3.1 Web references	17
Index	19

KEYBOARD

1.1 Reading Keyboard Scan Codes Through the PS/2 Interface on the XS board

1.1.1 Background

The PC/AT keyboard transmits data in a clocked serial format consisting of a start bit, 8 data bits (LSB first), an odd parity bit and a stop bit. The clock signal is only active during data transmit. The generated clock frequency is usually in the range 10 - 20 kHz. Each bit should be read on the falling edge of the clock.

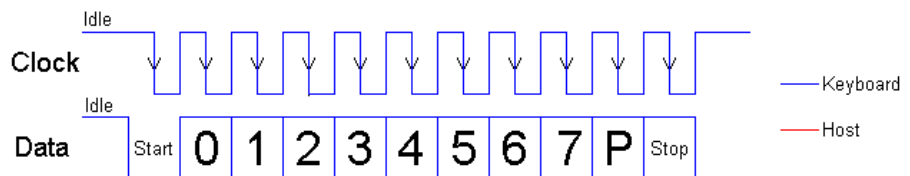


Figure 1 Keyboard transmission waveform

The above waveform represents a one byte transmission from the Keyboard. The keyboard may not generally change its data line on the rising edge of the clock as shown in the diagram. The data line only has to be valid on the falling edge of the clock. The Least Significant Bit is always sent first.

When a key is pressed, the keyboard transmits a 'make' code consisting of an 8-bit 'scan' code denoting the key pressed and, when the key is released, a 'break' code. The 'break' code (key released) consists of the same 8-bit scan code preceded by a special code-0F0H.

The keyboard handles combinations with control keys as SHIFT, CTRL, ALT, etc as two separate key presses, i.e., SHIFT-MAKE, 'A'-MAKE, SHIFT-BREAK, 'A'-BREAK. The 'A' scan code (1CH) is the same for both the shifted and unshifted state. To determine whether the 'A' scan code is interpreted as 'A' or 'a' the PC must keep track of the presence or absence of a prior SHIFT-MAKE.

Read more about the AT keyboard in Appendix A, “PC AT keyboard ref”.

1.1.2 Assignment

Create a circuit that accepts scan codes from a keyboard attached to the PS/2 interface of the Xstend Board. The binary pattern of the scan code is displayed on the bargraph LEDs. In addition, if a scan code for one of the '0'-'9' keys arrives, then the numeral will be displayed on the right LED display of the Xstend Board.

1.1.3 Problem definition

Here are the definitions listed that is needed to complete the design.

1.1.3.1 Port definitions

The inputs and outputs of the circuit as defined are as follows:

Table 1 Port interface description

name	Type	Range	description
kb_data	IN	std_logic	The scan code bits enters through this input.
kb_clk	IN	std_logic	The keyboard clock signal enters through this input.
sys_clk	IN	std_logic	The system clock used to clock the system.
db	OUT	std_logic_vector 7 downto 0	Drive the segments of the bargraph on the XS Board.
rsb	OUT	std_logic_vector 6 downto 0	Drive the segments of the right LED on the XS Board.
fcs	OUT	std_logic	chip select select to program the XS Board Set to '1' in VHDL code
rlcs	OUT	std_logic	chip select select to program the XS Board Set to '1' in VHDL code
rrcs	OUT	std_logic	chip select select to program the XS Board Set to '1' in VHDL code
sdcs	OUT	std_logic	chip select select to program the XS Board Set to '1' in VHDL code

1.1.3.2 Procedure definitions

Within the main body of the architecture section, these processes should be implemented:

sync_keyboard

Synchronize the external inputs to the system clock. Creates an internal copy of the keyboard signals that only changes values on the positive edge of the system clock.

detect_falling_kb_clk

Creates an output which is one during one system clock cycle when a falling edge of the synchronized kb_clk signal has been found. This can be done by comparing the current value of the synchronized kb_clk signal with the old value (stored in the previous clock period) of the synchronized kb_clk signal. Remember that the output signal from this process should not be used as a clock, as this would create a design with multiple clock domains.

convert_scancode

When edge_found is one will this process shifts the data bit on the kb_data input into the most significant bit of a 10-bit shift register. After 11 clock cycles, the lower 8 bits of the register will contain the scan code, the upper 2 bits will store the stop and parity bits, and the start bit will have been shifted through the entire register and discarded.

Table 2 Scan codes for the alpha numerics 0 to 9

KEY	SCANCODE	rsb
1	00010110	1101101
2	00011110	0100010
3	00100110	0100100
4	00100101	1000101
5	00101110	0010100
6	00110110	0010000
7	00111101	0101101
8	00111110	0000000
9	01000110	0000100
0	01000101	0001000
Else		0010010

The value in the shift register is inverted and applied to the segments of the LED bargraph. Since the bargraph segments are active-low, a segment will light

for every '1' bit in the shift register. The LED segment drivers are not stored in a register so there will be some flickering as the shift register contents change.

If the scan code in the shift register matches the codes for the digits 0-9, then the right LED digit segments will be activated to display the corresponding digit. If the scan code does not match one of these codes, the letter 'E' is displayed.

1.1.3.3 Tips and tricks

To avoid a lot of trouble in the design phase a few hints are needed

- Do not use multiple clock definitions, i.e. , use only one statement with 'EVENT defining a clock.
- The use of 'EVENT and 'LAST_VALUE are not useful in the context of synthesis except for defining the rising or falling edge of clock.

1.1.4 Interface

The synthesis tool needs to know how the FPGA is connected to the external world. A attribute description included in the the VHDL description will be used for this purpose. How to include this into the design will be demonstrated later.

```
type exemplar_string_array is array (natural range <>, natural range <>)
of character;
attribute pin_number : string;
attribute array_pin_number : exemplar_string_array;
attribute buffer_sig: string;
attribute buffer_sig of sys_clk : signal is "IBUFG";
attribute pin_number of sys_clk : signal is "P88";
attribute pin_number of fcs : signal is "P41";
attribute pin_number of rlcs : signal is "P79";
attribute pin_number of rrcs : signal is "P80";
attribute pin_number of sdcs : signal is "P132";
attribute pin_number of kb_clk : signal is "P31";
attribute pin_number of kb_data : signal is "P30";
attribute array_pin_number of db : signal is ("P67", "P60", "P62",
"P57", "P49", "P46", "P44", "P68");
attribute array_pin_number of rsb : signal is ("P48", "P42", "P27",
"P29", "P28", "P40", "P47");
```

This attribute description is prepared in a text file and can be copied from
/proj/tde/TSTE70/kursmaterial/Lab/Kbd/keyboard_ucf_attributes.txt

1.1.5 Requirements

Here are the requirements to pass the laboratory

- Implement the design using hdl designer

- Declare a symbol and corresponding VHDL view
- Design a test bench to verify the design
 - The testbench should be built as a structural top level, i.e. block diagram in hlldesigner, and instantiate the keyboard symbol as a component.
 - The testbench should also include a stimuli and a controll block in the same block diagram as the keyboard reader. The stimuli block generate the input data to the keyboard. The control block analyzes the design.
 - The testbench have to implement at least two different scancodes.
 - The testbench should indicate each testcase with correct or erraneous behavior in the transcript window of the simulator. Take a look at the the ASSERT statement in VHDL.
- Synthesize the design and demonstrate the function on the XS FPGA board.
- Summarize a number of key figures for the design
 - How much of the total available design space is used
 - What is the maximum clock frequency allowed for the design.

1.2 Design

It is here assumed the reader have a basic knowledge of hlldesigner. The experience in the hdl designer intro in this course should be sufficient.

An important difference is how to start hlldesigner. To enable one group working together on the same design, we prepared the setup. Follow the design below and the tool start and file permissions are set correctly.

1.2.1 Introduction

Your designs should preferrably be saved in a new library directory.

- 1 Start a mentorskal
- 2 Start hlldesigner using
renoirlab
- 3 Open the design manager project view.
- 4 Create a new library named KEYBOARD and set root directory to /proj/tde/labs/labgrupp<nn>/Lab. Replace <nn> with your group number.
- 5 Make the library default and open/explore the library. Continue the work in this library and complete the exercise. Either double click on the library name or use right mouse button to acces the popup menu for the library and choose explore.

1.2.2 Adding the pin attributes

The symbol pin attributes are necessary to the synthesis step. Here is the how to include the prepared pin attributes into the keyboard design. To be able to use the attributes a package reference is also needed.

- 1 Create the symbol for the design.
- 2 Open the symbol editor window
- 3 Locate "User:" under Declarations in the window. Place the mouse pointer over the text. Double-click using left mouse button on the text.
- 4 Open the pin attributes text file in a text editor.
- 5 Copy and paste the pin attributes from the text file into the declaratiосn dialog box.
- 6 Press Ok and the attributes should appear under User declarations.
- 7 Move the mouse pointer over the package list text. Press and release right mouse button. From the popup menu select "Package Refrences..."
- 8 Select library Exemplar from the Library list. In the packages window select exemplar_1164.
- 9 Press the Add button and then Ok
- 10 Save the symbol.
- 11 Finish the design with ports and behavioral view.

The symbol and interface is now complete. Continue to add the behavior of the design and validate the functionality. Here you may have use of the following.

Add stimulis to kb_clk and kb_data and validate the function. These can either be written into the Modelsim window or put in a macro file. The following example simulate the scancode for the '4' key (remember that the keyboard should only generate a clock when there are bits to send):

```
force -repeat 83 sys_clk 0 0, 1 42
force kb_clk 1 25us, 0 100us, 1 150us, 0 200us, 1 250us, 0 300us, 1
350us, 0 400us, 1 450us, 0 500us, 1 550us, 0 600us, 1 650us, 0 700us, 1
750us, 0 800us, 1 850us, 0 900us, 1 950us, 0 1000us, 1 1050us, 0 1100us,
1 1150us
force kb_data 1 25us, 0 75us, 1 175us, 0 275us, 1 375us, 0 475us, 0
575us, 1 675us, 0 775us, 0 875us, 0 975us, 1 1075us
run 1200 us
```

These commands can also be put in a macro file to simplify repeated tests. Use a text editor to create the file and put the commands to be executed in it. It is important to keep all text on each force command in one single line for each force.

1.2.3 Creating the testbench

Here we describe briefly the steps to create a testbench.

- 1 Create a new symbol for the design.
- 2 Open the symbol editor window
- 3 Open a new as block diagram
- 4 Instantiate the keyboard reader as a component. Press the green block entry in the toolbars menu, i.e. , "add component".
- 5 Now a window with available components appear. Drag and drop the keyboard symbol on the testbench block diagram window.
- 6 Continue and add the stimuli block and control block to the schematic.
- 7 Add behavior to the control and stimuli blocks.
- 8 Validate the testbench design. Important now is to choose generate and compile through component
- 9 Validate the design

1.3 Synthesis flow (using LeonardoSpectrum).

1.3.1 Synthesize the Design

The design flow buttons include several steps in one command. There are flows defined for simulation and synthesis. The synthesis assume a validated and compiled design.

- 1 Select the Keyb_reader design unit in the design browser
- 2 Start synthesis by choosing the synthesis flow button. You find this next to the compile and generate toolbar buttons.
- 3 The LeonardoSpectrum Synthesis settings appear. Make sure to change the following in the setup.

Technology: FPGA/Xilinx/Spartan2
Device: 2s50tq144
Speed Grade: -6
Design frequency: 25 MHz
Run Options: Run Integrated Place and Route should selected

- 4 Press ok. The synthesis tool is now started.
- 5 Accept the LeonardoSpectrum Invoke Settings. Choose LeonardoSpectrum Level 3 as license selection.

- 6 The synthesis now start and generate netlist.
- 7 The tool also produce a bitfile to download on the XS Board.
- 8 Locate the output edf netlist file. In the same library the bit file is located.

1.3.2 Results

A lot of reports has again been generated. Answer the following questions:

Device Utilization for 2s50tq144

Resource	Used	Avail	Utilization
IOs	92%
Function Generators	1536%
CLB Slices	768%
Dffs or Latches	2082%

Clock : Frequency

clk : MHz

1.4 Downloading the Design

The steps for downloading the bit-file and testing the design using an Spartan2 development card combined with an XStend Board are as follows:

- 1 Put the generated bit file on diskette
- 2 Go to the lab computer with the XS Board
- 3 Start the graphical tool GXLOAD
- 4 Drag the file from diskette to the bit file window.
- 5 Press download.
- 6 Press keys on the keyboard and observe the results on the LED displays.



PC AT KEYBOARD REF

A.1 Description of keyboard

The original keyboard design had a single chip microprocessor, but now a customised controller chip is used. This keyboard controller chip takes care of all keyboard matrix scanning, key de-bouncing and communications with the computer, and has an internal buffer if the keystroke data cannot be sent immediately. The PC motherboard decodes the data received from the keyboard via the PS/2 port using interrupt IRQ1.

The one thing that these keyboards do not generate is ASCII values. With a typical AT keyboard having more than 101 keys, a single byte could not store codes for all the individual keys, plus these keys along with shift, control, or alt, etc. Also for some functions there is no ASCII equivalent, for example 'page up', 'page down', 'insert', 'home', etc.

When the keyboard controller finds that a key is being pressed or released it will send this keystroke information, known as scan codes, to the PIC microcontroller. There are two different types of scan codes - make codes and break codes.

The communications protocol is bi-directional, but here we only discuss the keyboard to host part.

A.1.1 make code

A make code is sent whenever a key is pressed or held down. Each key, including 'shift', 'control' and 'alt', sends a specific code when pressed. Cursor control keys, 'delete', 'page up', 'page down', 'ins', 'home' and 'end', send extended make codes. The make code is preceded by 'EO'h to indicate an

extended code. The only exception is the 'pause' key that starts with a unique 'E1'h byte.

A.1.2 break code

A break code is sent when a key is released. The break code is the make code preceded by 'FO'h byte. For extended keys the break code has an 'EO'h preceding the 'FO'h and make code value. The only exception is the 'pause' key as it does not have a break code and does not auto-repeat when held down.

A.1.3 key code

Every key is assigned its own unique code so that the host computer processing the information from the keyboard can determine exactly what happened to which key simply by looking at the scan codes received. There is no direct relationship between the scan code generated by a particular key and the character printed on the key top.

The set of make and break codes for each key comprises a scan code set. There are three standard scan code sets -numbered 1, 2, and 3 - stored within the keyboard controller. Scan code set 1 is retained for compatibility for older IBM XT computers. Scan set 3 is very similar to the set 2 but the extended codes are different. Scan code set 2 is the default for all AT keyboards and all scan codes discussed here are from this set.

A.1.4 scan code

If, for example, you press 'shift' and 'A' then both keys will generate their own scan codes, the 'A' scan code value is not changed if a shift or control key is also pressed. Pressing the letter 'A' generates '1C'h make code and when released the break code is 'FO'h, '1C'h.

Pressing 'shift' and 'A' keys will generate the following scan codes :

The make code for the 'shift' key is sent '12'h.

The make code for the 'A' key is sent '1C'h.

The break code for the 'A' key is sent 'FO'h, '1C'h.

The break code for the 'shift' key is sent 'FO'h, '12'h.

If the right shift was pressed then the make code is '59'h and break code is 'FO'h, '59'h.

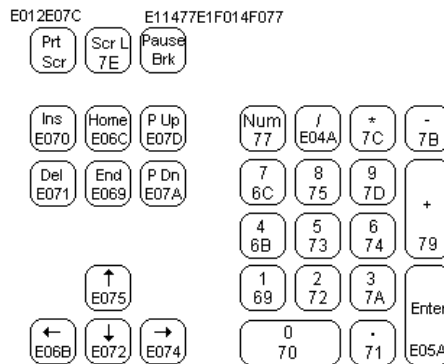
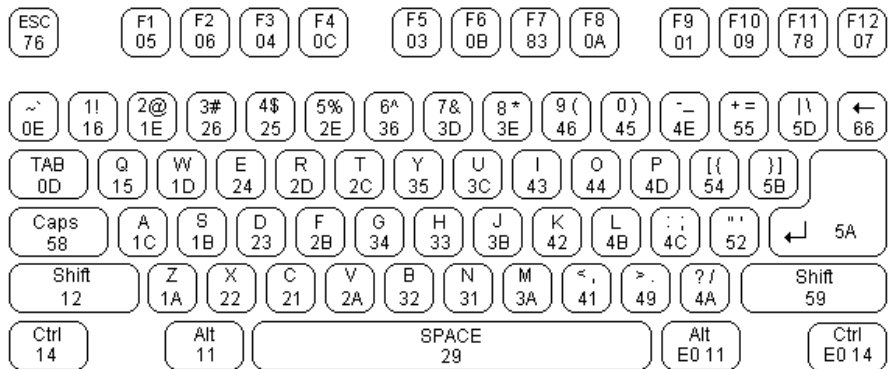
By analysing these scan codes the PC software can determine which key was pressed. By looking at the shift keystroke the software can distinguish between upper and lower case.

A.1.5 Keyboard serial data

The AT keyboard transmission protocol is a serial format, with one line providing the data and the other line providing the clock. The data length is 11 bits with one start bit (logic 0), 8 data bits (lsb first), odd parity bit and a stop bit (logic 1). The clock rate is approximately 10 to 30kHz and varies from keyboard to keyboard.

A.2 Scancodes and commands

A.2.1 Keyboard layouts with codes



A.2.2 Scan code tables

Table 1.1 Keyboard alpha numeric scan codes

key	make	break	
A	'1C'h	'FO'h	'1C'h
B	'32'h	'FO'h	'32'h
C	'21'h	'FO'h	'21'h
D	'23'h	'FO'h	'23'h
E	'24'h	'FO'h	'24'h
F	'2B'h	'FO'h	'2B'h
G	'34'h	'FO'h	'34'h
H	'33'h	'FO'h	'33'h
I	'43'h	'FO'h	'43'h
J	'3B'h	'FO'h	'3B'h
K	'42'h	'FO'h	'42'h
L	'4B'h	'FO'h	'4b'h
M	'3A'h	'FO'h	'3A'h
N	'31'h	'FO'h	'31'h
0	'44'h	'FO'h	'44'h
P	'4D'h	'FO'h	'4D'h
Q	'15'h	'FO'h	'15'h
R	'2D'h	'FO'h	'20'h
S	'1B'h	'FO'h	'1B'h
T	'2C'h	'FO'h	'2C'h
U	'3C'h	'FO'h	'3C'h
V	'2A'h	'FO'h	'2A'h
W	'1D'h	'FO'h	'1D'h
X	'22'h	'FO'h	'22'h
Y	'35'h	'FO'h	'35'h
Z	'1A'h	'FO'h	'1A'h
1	'16'h	'FO'h	'16'h
2	'1E'h	'FO'h	'1E'h
3	'26'h	'FO'h	'26'h
4	'25'h	'FO'h	'25'h
5	'2E'h	'FO'h	'2E'h
6	'36'h	'FO'h	'36'h
7	'3D'h	'FO'h	'3D'h
8	'3E'h	'FO'h	'3E'h
9	'46'h	'FO'h	'46'h
0	'45'h	'FO'h	'45'h

A.3 Further reading

A.3.1 Web references

Interfacing the AT keyboard

<http://www.beyondlogic.org/keyboard/keybrd.html>



Index



