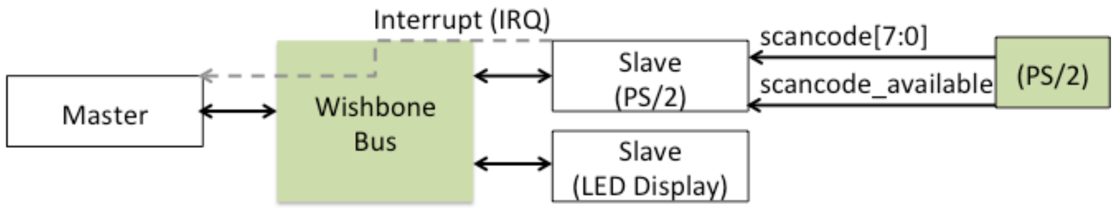
ECE 4401 (Fall 2014)
Lab 5 – PS/2 Keyboard Controller

Specification:

The basic goal of this lab is to design logic to control a PS/2 keyboard. The project specifications are as follows:

Connect a PS/2 keyboard to the PS/2 port on your board. The seven-segment display will display the last 4 characters pressed on the keyboard and the small LEDs will display the ASCII representation of the last character pressed.



You may use the Wishbone LED display module and Wishbone interface from previous lab. You will also need a PS/2 keyboard wishbone module that takes bits from the serial PS/2 port and creates 8-bit scancodes. When it receives a scancode, it will be extended to place an interrupt on the wishbone bus. You will use the IRQ signals in wb_intercon.vhd to implement interrupts. You will create a master reader module that waits for the PS/2 interrupt, and when it gets it, it will read the scancode from the PS/2 module. The act of reading the scancode should clear the interrupt. There is no explicit interrupt acknowledge signal on the Wishbone bus.

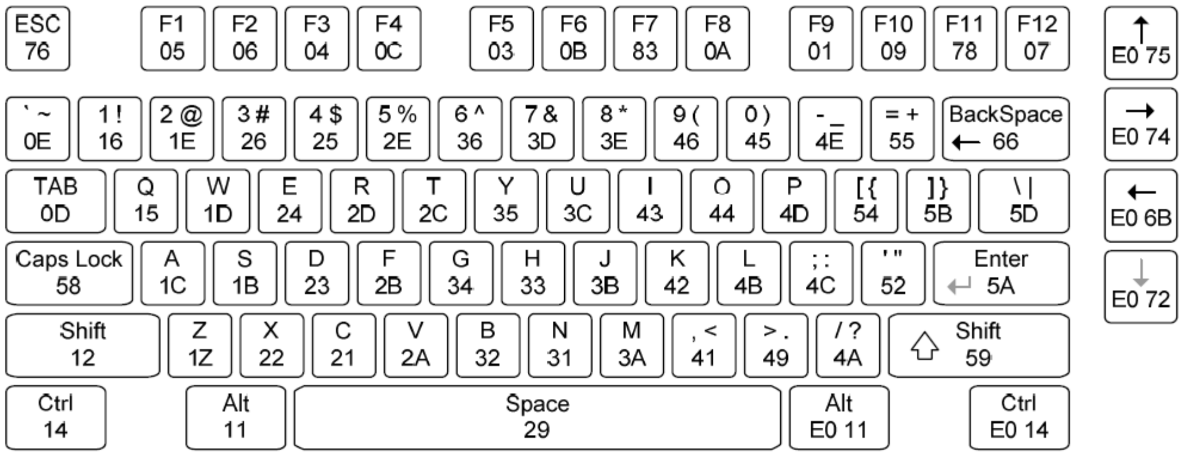
The master reader module will take the scancode and keep three flags that indicate whether the ctrl, shift, or the alt key is pressed. Based on the status of these three flags and the current scancode, the scancode2ascii module (that is provided to you) will create an ASCII code. These ASCII codes will then be written to a wishbone interface to the wishbone display module that displays data on the 4x7 segment display.

A period '.' should be displayed as the dot next to the previous character not as the dot after the next character. In other words the sequence 'A', 'S', 'D', '.', 'F' should display ASD.F on the LED display, not SD .F. This code is provided for you in the string2leds module. This module is passed the last eight characters that have been entered and will output the appropriate segment and digit bits. Therefore, the wishbone display module will need to take in ASCII characters from the reader and hold onto the last 8 characters and present them to the string2leds module.

The display module is also responsible for handling the backspace key, which should delete the last character from the LED display and shift all the characters to the right.

Your keyboard controller should be able to handle the standard typewriter keyboard but does not need to handle extended keyboard features like cursor keys, numeric keypad or function keys.

The keyboard scancodes are as follows:



The ASCII table is given below:

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [	5c \	5d]	5e ^	5f _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c |	7d }	7e ~	7f del

Background:

On each key press, the keyboard will generate 3 scancodes that will be sent to the reader module. The first code is known as the "make code" and it is generated on a key press. The next two scancodes are generated one after the other on a key release and together are known as "break code". For example, if you press "a", it will generate a make code of x"1C" and on key release will generate a break code of x"F0" and x"1C". You need to make sure that the reader module only displays "a" once. This means that the make code will be displayed while the break code will not be displayed.

Also, the flags indicating the key press of ctrl, shift or alt will be set when the respective make code (x"12", x"11", and x"14" for left- ctrl, shift, and alt respectively) is generated and will be reset when the respective break code is generated. As you have to keep track of the fact that a key has been released, you should keep a status flag for a key release event i.e. when you receive an x"F0" the status flag is asserted while it is de-asserted when you receive any other scancode.

Note that when a key is pressed continuously, it would generate multiple make codes. In this case, your display module should display the character multiple times. For example, if "a" is pressed continuously for a longer duration, the keyboard will generate multiple x"1C" make codes. All these generated codes should be displayed. On key release, a single break code will be generated and that's where you should stop displaying the character.

Notes:

- The library modules you need for this lab are,
  - clock_divider
  - wb_intercon
  - wb_arbiter
  - wb_segled
  - kb_interface
  - scancode2ascii
  - string2leds (which uses char2led and char_led_control)

- Modules you need to design:
  - **wb_ps2_kb module**: This module interfaces with the kb_interface module which receives the scancode from the keyboard. You will design a state machine that interacts with the master module through the wishbone bus. It will interrupt the master when it has a valid scancode to transmit and the initiate the data transfer, once the master is ready to receive it.

  - **Reader** (Master module): This module is the only master in the system. It will wait for the interrupt from the keyboard interface module. Upon getting an interrupt, it will assert the CYC_O and STB_O signals to read the scancode data from the wb_ps2_kb module. When it gets the scancode, it will then pass that to the scancode2ascii module to convert it to an ascii code. After that, it will send the ascii code to wb_leds and display module.

  - **Display** (Slave Module): This module will wait for the ascii code from the master. Once it gets it, it will display it on 7segment LEDs through the string2leds module. This module is responsible for storing the last 8 characters entered through the keyboard. It also needs to take care of the situation when 'backspace' is pressed i.e. delete the character displayed on the right most 7-segment display and fill the left most character register with zeros.