ENGN3213 Assignment 1
Technical Documentation

1 Overview

The board used for this assignment is the BASYS 3. This board has 4 seven-segment displays meaning that words must have a maximum of 4 letters.

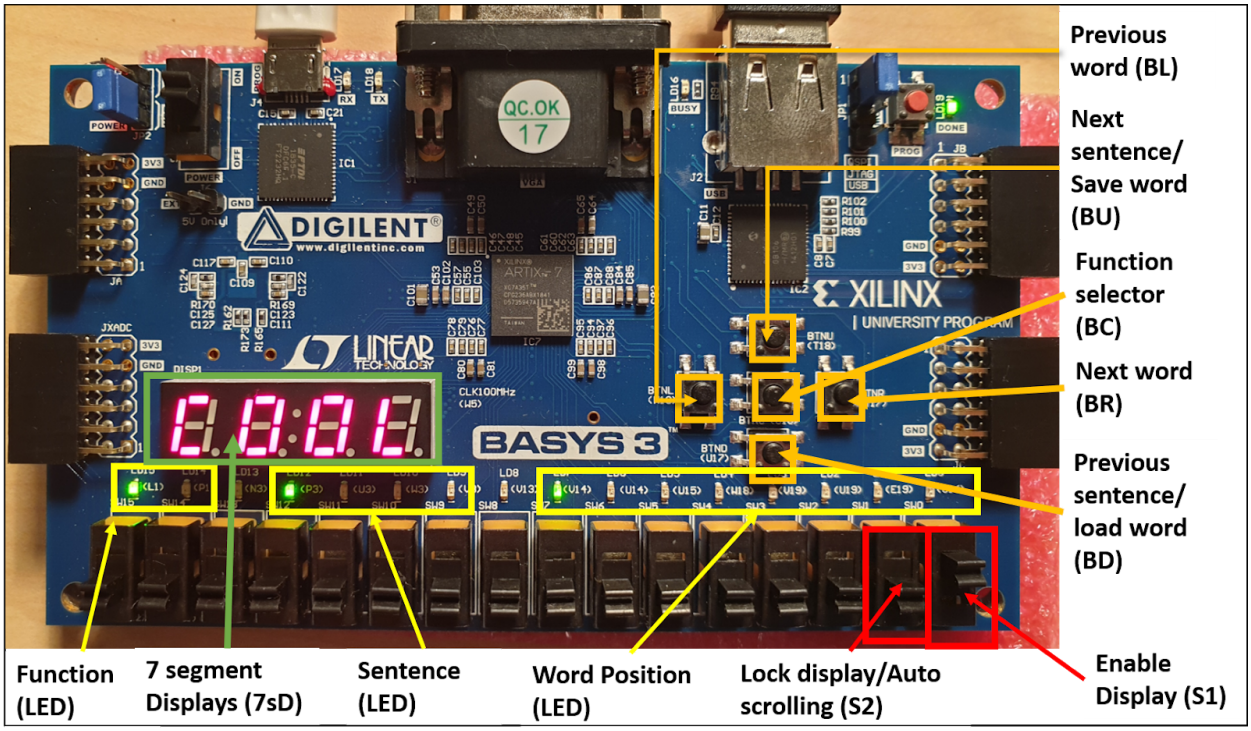
Three different functions are available. All functions are integrated together in a single design and can be switched between. The active function is displayed. The display can be turned off with a slide switch.

Function 1: The most recent key pressed is shown on the seven-segment display, or it is blank if the key is unrecognised. Updating the seven-segment display can be disabled with a slide switch.
Function 2: Four preset sentences can be displayed. These sentences can be switched between. Words within a sentence can be switched between, or can be automatically scrolled through, activated with a slide switch. The current sentence and word are indicated using the onboard LEDs.
Function 3: The user can type in characters which will be added to the current word, up to a maximum of four characters. The user can save changes to the current word and scroll between words. Unsaved changes are lost when scrolling between words. Any of the four preset sentences can be loaded and edited freely.

2 User Interface

The user interface can be seen in Figure 1 below.

Fig. 1: User interface diagram



There are three different modes, corresponding to each of the three functions which the user can choose from. The function that the user is currently in is displayed by the function LEDs. The user can switch between modes by pressing the function selector (BC).
For all modes the enable display slide switch (S1) must be on for the 7 segment display to be enabled.

2.1 Mode 1

This first mode displays last key pressed on the keyboard. If key pressed is not listed in the decoder (see appendix 1) then nothing will be displayed. The lock display switch (S2) can be used to toggle when new inputs will be received. Only when it is on can the character displayed be updated.

2.2 Mode 2

This mode is able to run sentences with up to 8 words. 4 example sentences have been provided (see appendix 2). When the auto scrolling switch (S2) is on, the device will automatically change to the next word after 1 second. The user can manually move between words using BR and BL to move to the next or previous word respectively. This can also be done using the right and left arrow keys of the keyboard. The sentence will restart if the next word is selected after the last word and the last word of the sentence will be displayed if the previous word is selected.

The user can manually move between the 4 example sentences by pressing BU or BD or the corresponding up and down arrow keys on the keyboard. Note: the sentence position is maintained when changing so if the 3rd word is displayed when the sentence is changed the 3rd word of the next sentence will be displayed.

2.3 Mode 3

This mode allows the user to write and edit words with the opportunity to save or load them to form sentences of up to 8 words.

Editing words:

This board will display a flashing | to indicate the cursor position to the user. When a recognised key is released the flashing display will update to display that key and move to the next display. Once all 4 displays are set (cannot see cursor), no further keys will be accepted. The user can remove letters by pressing backspace.

Moving between words:

Similar to mode 2, pressing BR or BL (or right or left arrow keys) will move to the next or previous word. Like mode 2, there are also the same 8 LEDs to display the current word's position in the sentence. There is no auto scrolling in this function.

Saving and Loading:

Once the user has completed a word they can save the word by pressing BU or the up-arrow key. This commits the new word meaning they can return to it after changing other words. If they move to the next or previous word their changes will not be remembered.

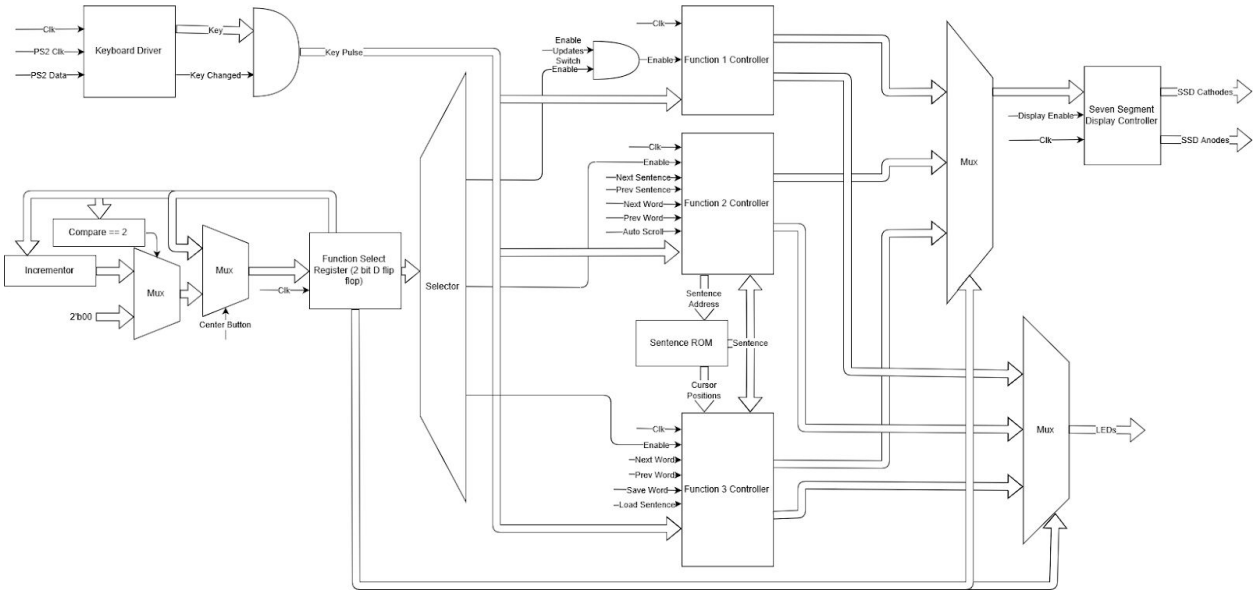
Pressing BD or the down-arrow key will cause the last used example sentence from mode 2 to be loaded and able to be edited.

3 Technical Information

3.1 Top Level Structure

The top-level module contains six main sub-modules; the three functions, the keyboard driver, the preset sentence ROM and the seven-segment display controller. It also contains the logic for switching between the three functions as well as debouncers and edge detectors for the five buttons. Figure 2 shows a block diagram of the top-level module. The debouncers and edge detectors are not shown. All inputs to each function are labelled. To find which buttons they are connected to, see section 2; User Interface. The control inputs for each function only refer to the buttons, keyboard controls are implemented within each module using the Key Pulse bus. A breakdown and further details of the implementation of each of the functions can be found in the following subsections. Note that all registers and flip flops are clocked. Some do not have the clock connection shown, particularly in more complex figures. This was done to simplify diagrams.

Fig. 2: Modular structure of project



Function selection is implemented using a 2-bit counter. The counter increments each time the center button is pressed and is reset to zero once reaching two, limiting it to three values which are uniquely associated with each function. The counter is passed through a demultiplexer (labelled as Selector in Figure 2) to enable a single function module at a time. The counter also acts as the selection signal for the output multiplexers.

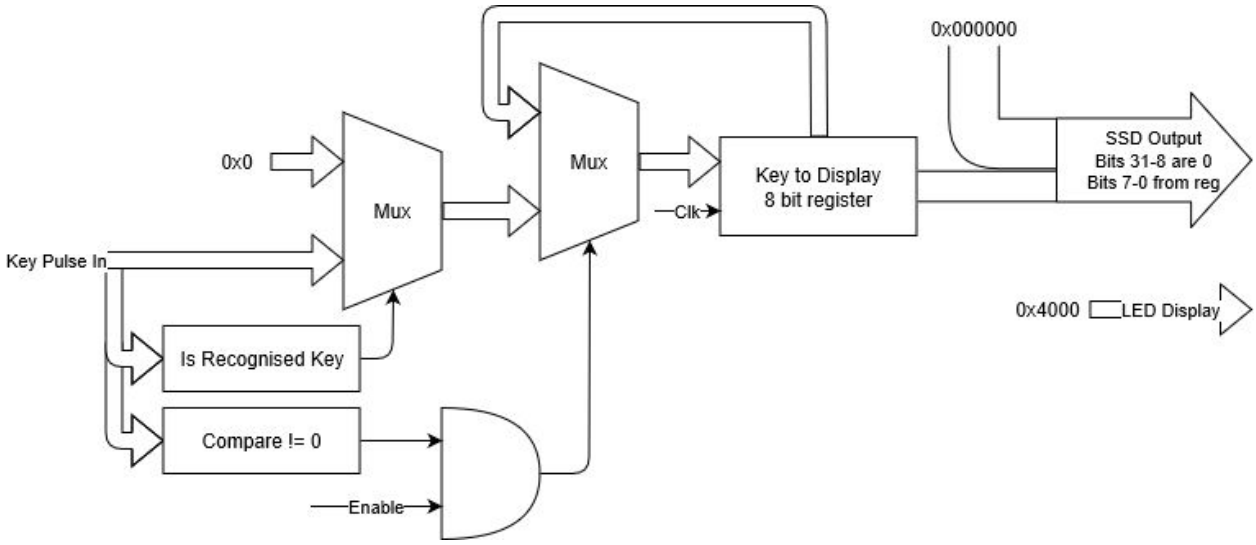
Each function module has an enable input. When this input is low the module should not respond to its other input signals. The output the function modules are identical. Each function has a display output bus containing the characters to display and an LED output bus containing the data to show on the LEDs. As such, function specific displays can be implemented in the function modules without needing an additional module to interpret different outputs for each function and combine them to make the final output display. This reduces the hardware cost of the design.

The keyboard module outputs the most recently pressed key and a signal which pulses each time a key is pressed. The pulse is combined with each bit of the key bus to create a signal which pulses the key code for one clock cycle whenever a key is pressed and is zero otherwise. Combining these signals eliminates an input to each function, thus simplifying the design.

3.2 Function 1 Controller

The function 1 controller takes an enable signal and the key pulse. Figure 3 shows a block diagram of function 1. The block ‘Is Recognised Key’ is shown in Figure 4.

Fig. 3: Function 1 Controller Breakdown

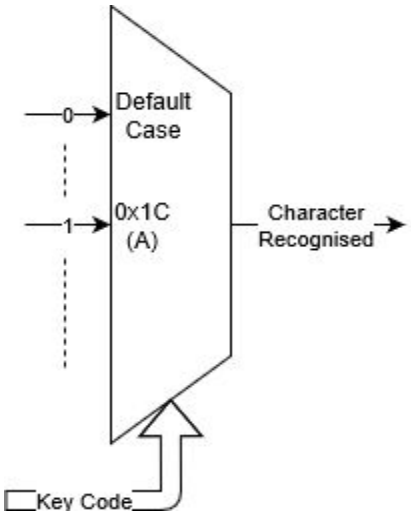


This controller sets its internal register containing the code of the character to be displayed if it is enabled and the key pulse is not equal to zero, indicating a key has been pressed. If the key is recognised, then the code is selected, otherwise zero is selected.

An alternative approach is to simply set the register to the key code whenever a key is pressed and design the seven-segment display module to not display unrecognised codes. However, this does not allow the display module to use unused codes to hold ‘meta characters’ such as a cursor which is used in function 3. Thus, the first approach was used to decouple the inputted code from the characters being displayed.

Since this function does not use any additional LEDs, the LED output is hardcoded to show the function.

Fig. 4: Is Key Recognised Signal



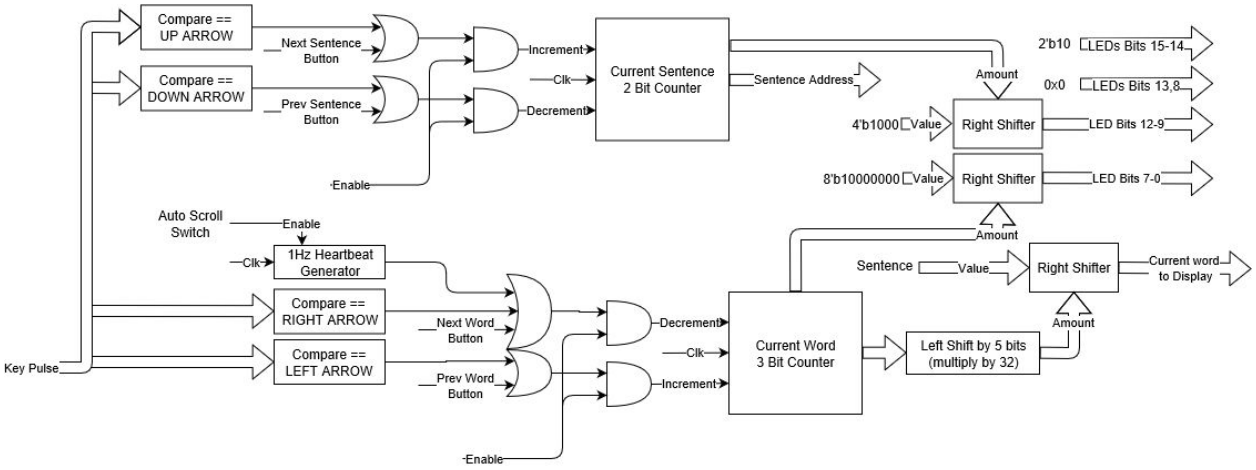
The is key recognised block is simply a multiplexer which selects from 2^8 inputs. Those inputs which correspond to recognised codes are set to one, those which don't are set to zero. A full list of recognised codes can be found in appendix 3.

3.3 Function 2 Controller

The function 2 controller contains two counters. These counters control the sentence and the word within the sentence being displayed. The sentence counter outputs its value onto a bus which is connected to the address input of the sentence ROM. It reads the selected sentence from the ROM and performs the shifting operations to display the current word.

An important note is that the first word in a sentence is considered at index 7 and the last at index 0. Hence, switching to the next word decrements the counter and switching to the previous word increments the counter. This choice was made as it simplifies the shifting logic to extract a word from a sentence since the first word is in the most significant 32 bits of the sentence and hence must be shifted right the most.

Fig. 5: Function 2 Controller Breakdown



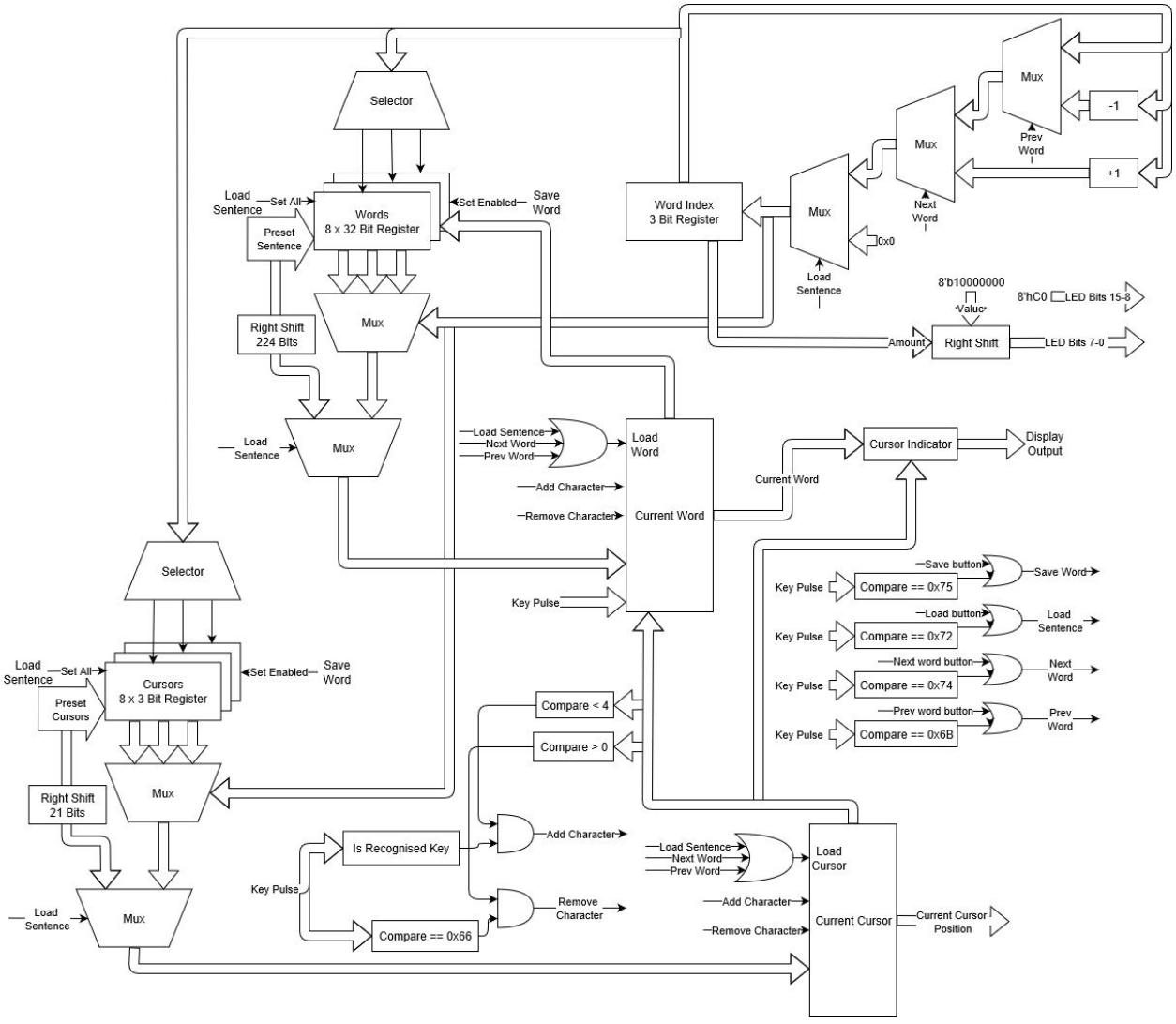
3.4 Function 3 Controller

There are four main components in the function 3 controller. The first is a pair of register banks, one which stores the words in the custom sentence, and one which stores the cursor positions for each word. These are register banks instead of a single large register as this allows individual words to be altered more easily. Individual registers can be selected for data to be latched in when saving the current word or, alternatively, they can all be enabled at once to load

a preset sentence. A multiplexer controls which register is outputting its word or cursor position. Figure 6 only shows three registers per bank for clarity.

This multiplexer is controlled by the second main component; the word selector. This consists of a register holding the index of the word currently being edited. This register can be incremented to move to the next word, decremented to move to the previous word or reset when a sentence is loaded. This is implemented using a series of nested multiplexers and adder units. These above two components can be seen in Figure 6.

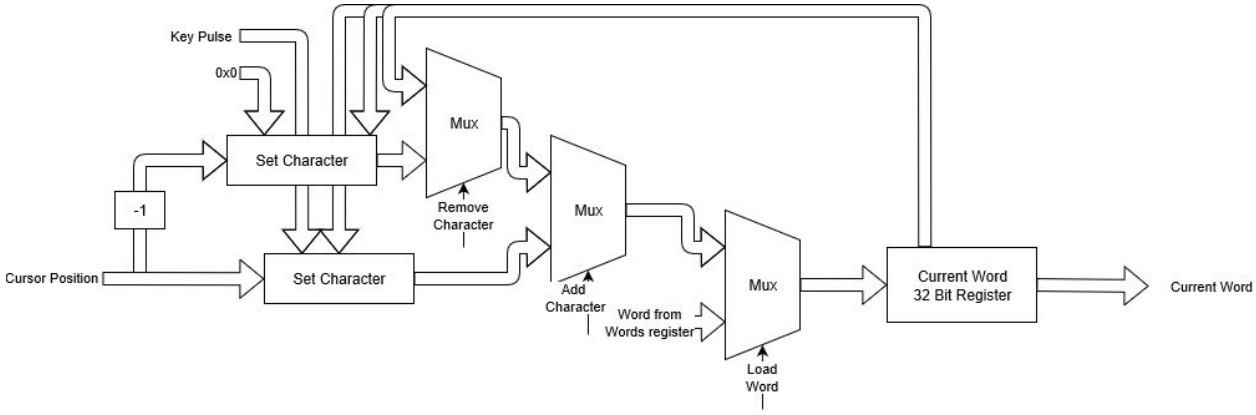
Fig. 6: Function 3 Controller Breakdown



The third component is labelled ‘Current Word’ in Figure 6 and a block diagram is shown in Figure 7. This block is responsible for updating the current word. It sets characters when characters are added or removed and outputs the current word to be displayed. Scrolling between words is implemented by loading the register with a new value. The word index register

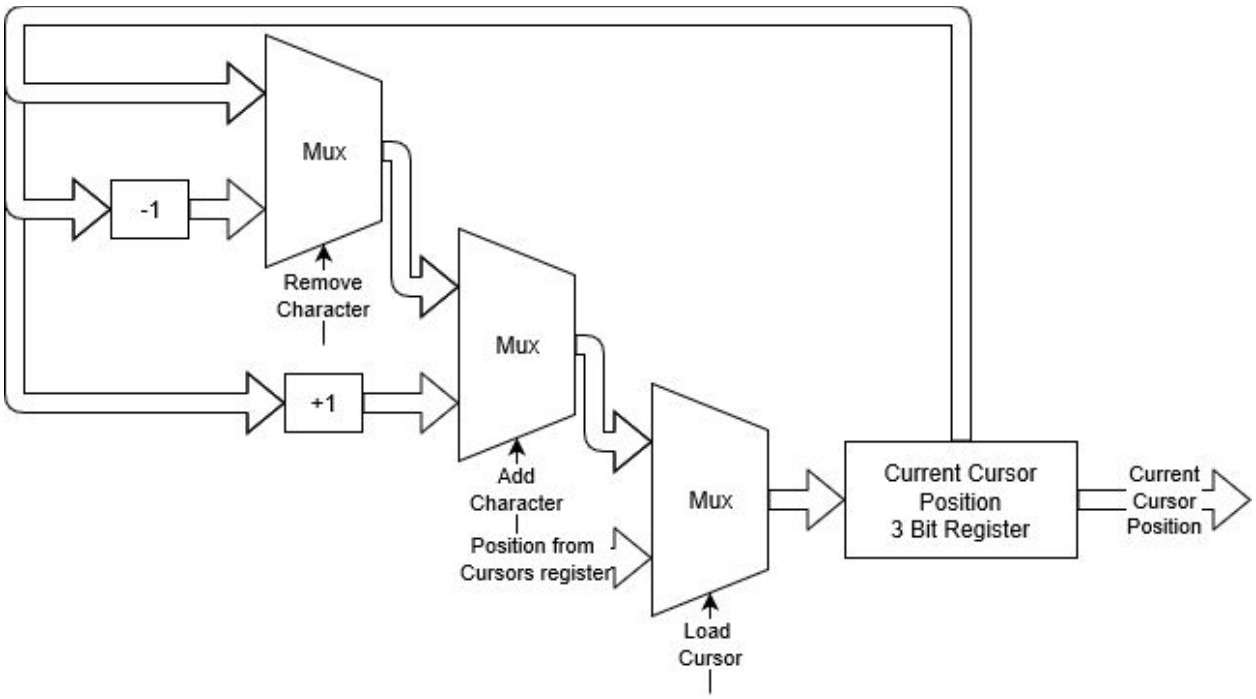
is used to select the word which is to be loaded. Further description of the set character block is below.

Fig.7: Current Word Block



The final component, ‘Current Cursor’ simply tracks the position of the cursor when editing a word. The internal register is incremented when a character is added, or decremented when a character is deleted. It also allows a new value to be loaded when a word is switched, and outputs it’s value so that the ‘Current Word’ block can correctly place a character or the cursor position can be saved. This block is shown in figure 8.

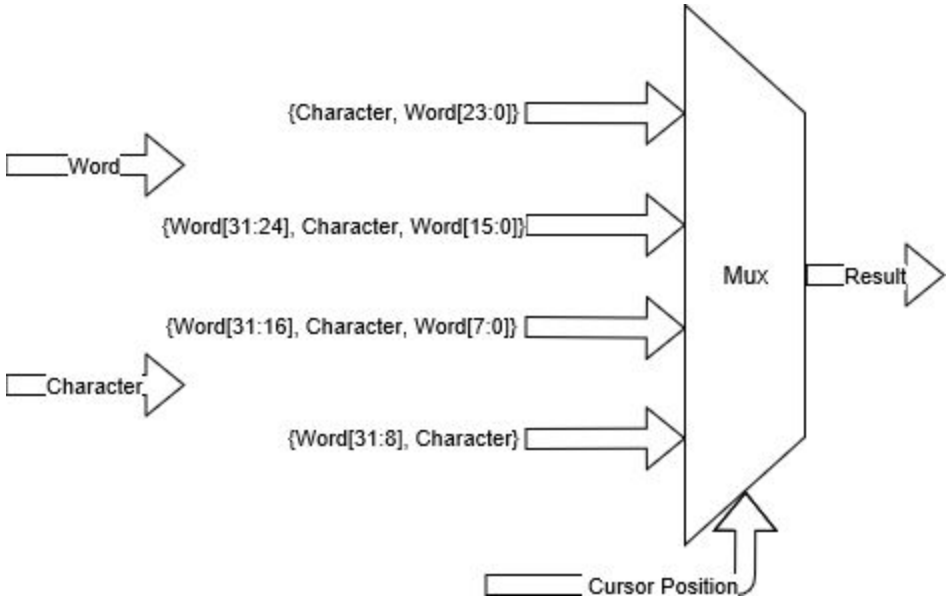
Fig. 8: Current Cursor Block



The set character block used to update the current word when a character is added or removed uses purely combinational logic. It takes a word, a character and a position and outputs a new

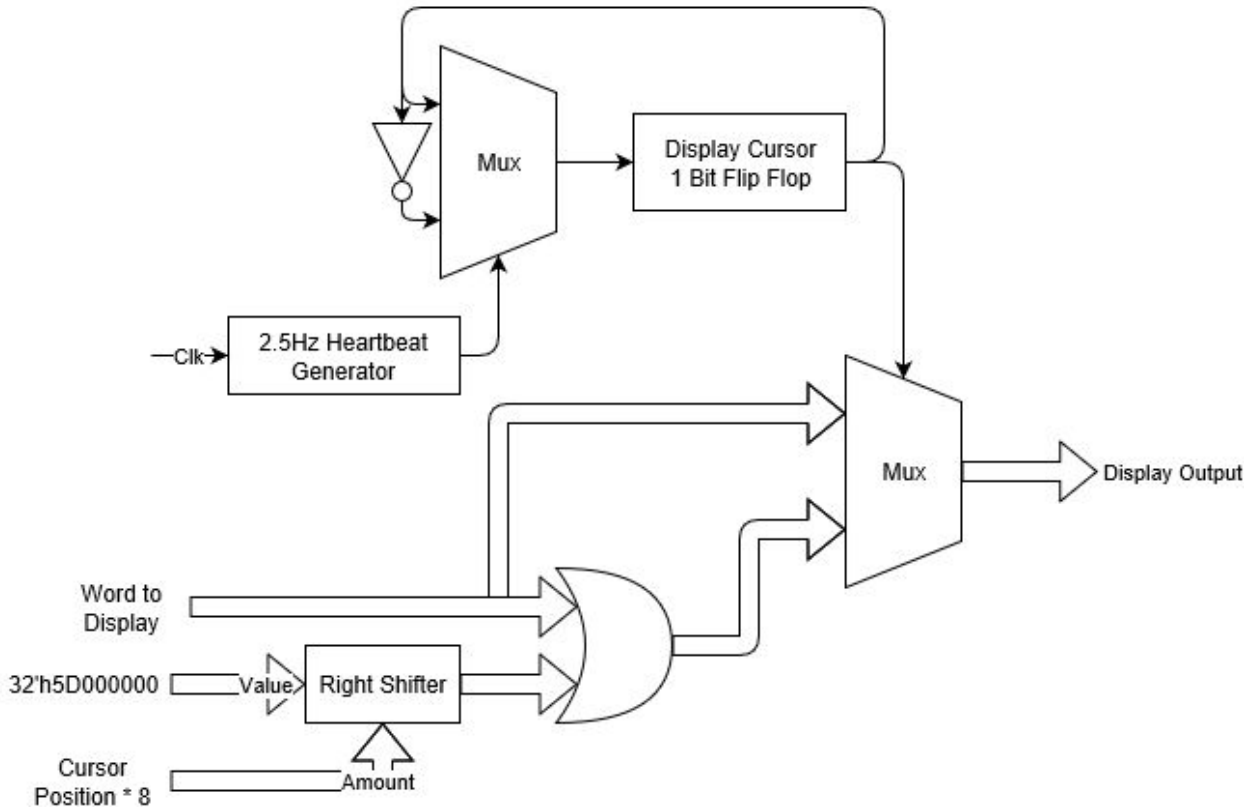
word with the specified character in the specified position. Each position is 8 bits long. This block is shown in figure 9.

Fig. 9: Set Character Block



as

Fig. 10: Cursor Indicator

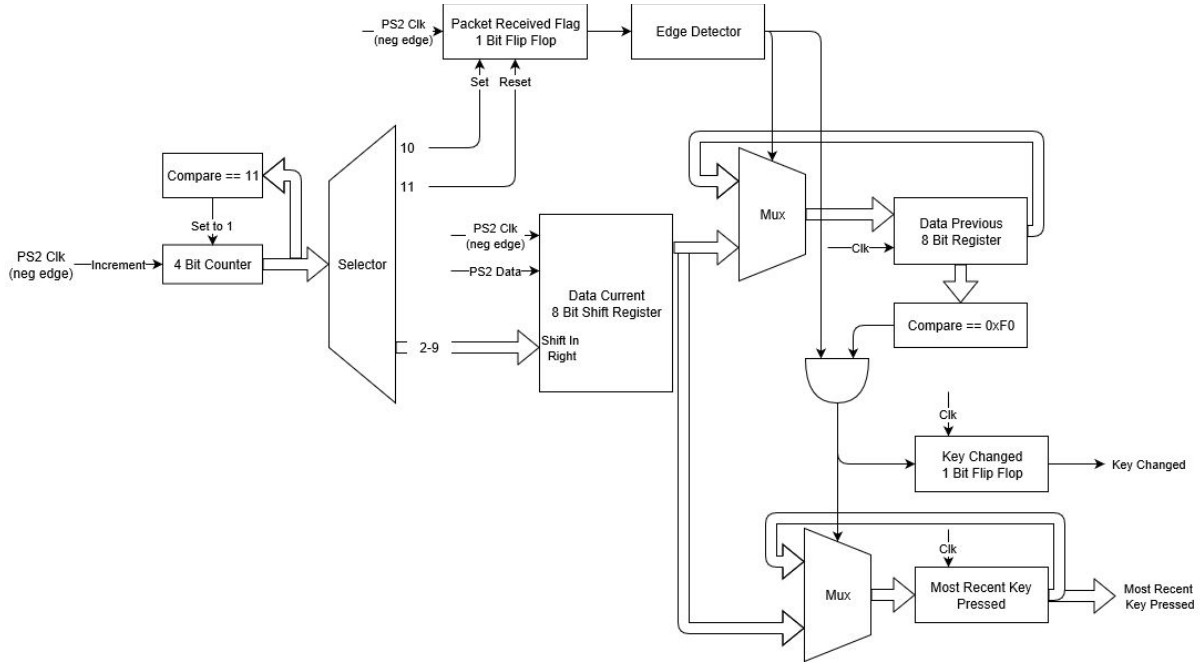


The cursor indicator simply positions a flashing bar at the cursor position. This is performed using a toggle flip flop which toggles every 0.4 seconds. This acts as a selector for a multiplexer which chooses to display the current word or the current word with a value shifted into the cursor position. The cursor ‘meta character’ is given the code 0x5D and is shifted to the first position of the first empty character. As empty characters are stored as 0x00, perform an OR operation sets the 8 bits of the character to 0x5D. Alternating between displaying this and not gives the effect of a blinking cursor.

3.5 Key Driver

The key driver adapted code from the IIT Kampur Electronics Club using their serial to parallel converter. This was done because the chosen code was simpler and error free during project development. The key driver accepts inputs of PS2Clk and PS2Data as well as the clock cycle (clk). The outputs are a 7-bit register, key, and wire key change.
The module uses a counter (b) to count each negative edge in the keyboard clock (PS2Clk). This assigns the corresponding data (PS2Data) into a demultiplexer filtering for an 8-bit shift register. The demultiplexer also sets a flag at the completion of each input cycle used to drive multiplexers to update the previous data register. Components outside of the multiplexor use the clock cycle input to synchronise. On conditions where key released the next corresponding filtered data packet is sent in a parallel pulse with the key changed signal.
Note: The output signal is the keyboard signal immediately following the release of the key. This approach was chosen because additional packets may be sent at, or during the pressing of the key while only one packet (code for that key) is sent after its release. For the purposes of this report keys being pressed and released have been used interchangeably.

Fig. 11: Key Driver structure breakdown

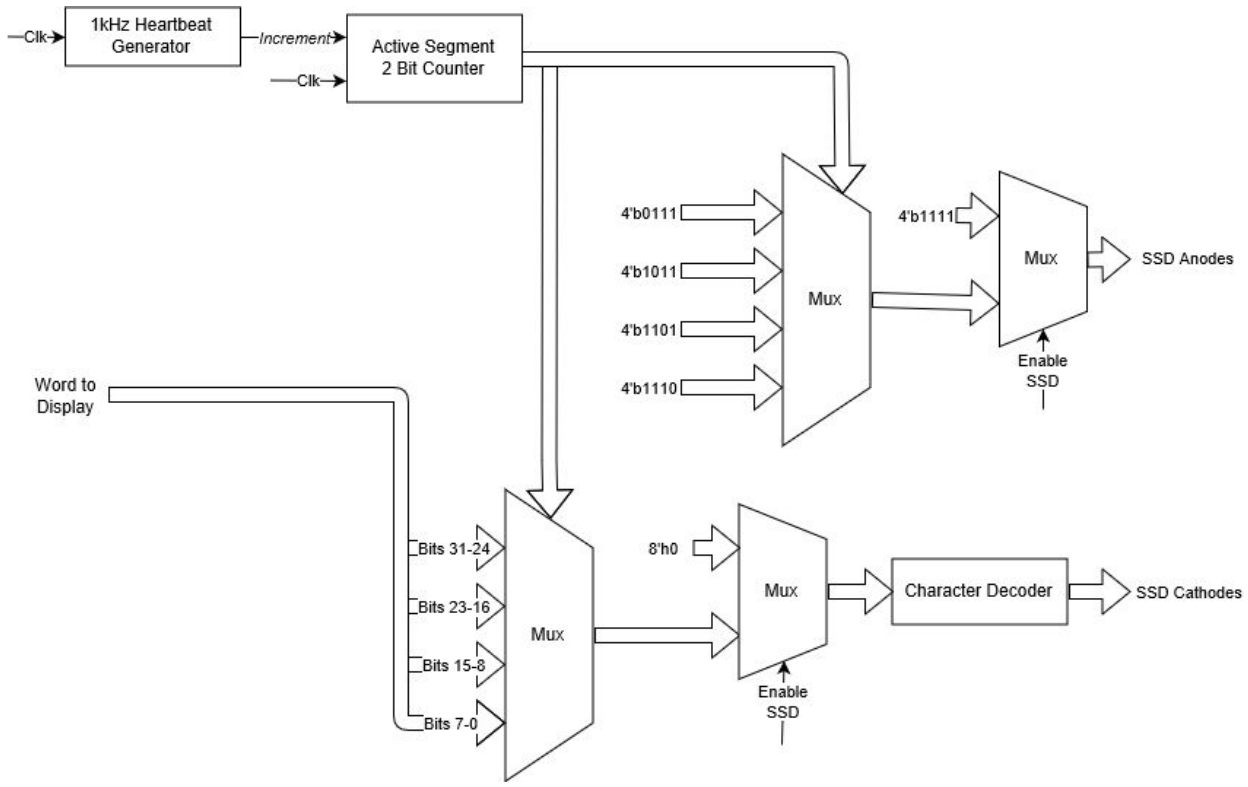


Note 2: The type of keyboard used can affect the effectiveness of the driver. Some keyboards were found to be not compatible while a few others required multiple keypresses.

3.6 Seven-segment Display Controller

The SSD Controller is used by all functions and is listed as the seven-segment display controller in figure 2. A scanning display is implemented, using a 1kHz heartbeat triggering a counter to iterate through each individual display. This counter selects the anode to ground, enabling that single display and selects the character from the output word to display. Enable switches a multiplexer between its current display value and 0 when the display is off. To disable the display, a high voltage value is asserted on each anode line, effectively disabling all the displays.

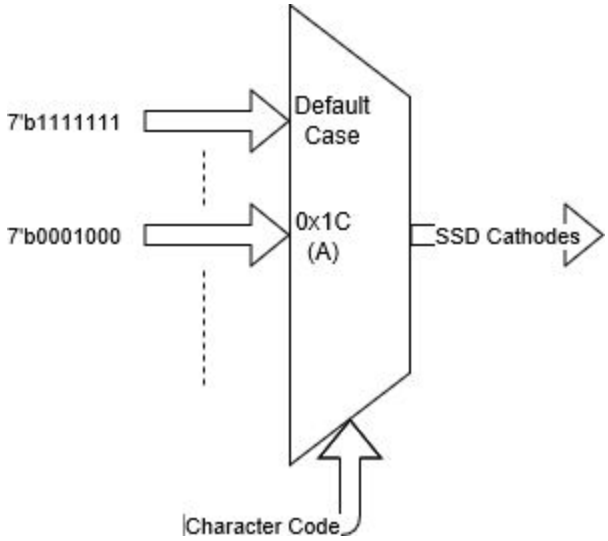
Fig. 12: SSD Controller Block



3.6.1 Decoder (character decoder)

The decoder module converters its 8-bit charCode input, which is the keyboard code for keys, into the corresponding 7-bit signal used to display the keys. The module uses a multiplexor for case matching of all the hexadecimal charCodes to segments code (inverse 7-bits setting value for each led), on a 1-1 basis.

Fig. 13: Decoder module as multiplexor



3.7 Sentence ROM

The sentences ROM module contains the predefined sentences (see appendix 2) that can be loaded for function 2 or 3. It also contains the lengths of each word in each sentence, which is used to set the cursor position when loading a sentence into function 3. The input is a 2-bit address bus used to select the sentence. The design is a simple asynchronous design implemented with multiplexers as explained in 3213 lectures.

3.8 Miscellaneous Modules

- Heartbeat generators are simple counter based designs as provided for 3213 test labs.
- The debouncer modules with clock dividers and edge-detector submodules are also provided in 3213 test labs.
- All components used to display or receive information have been added to the constraint file.

4 Source Code

Use Keyboard_Interface_TOP.v as the top-level module capable of all functions. The functions have been integrated in such a way that they cannot built as standalone functions.

5 References

Basic reference information:

"Basys 3 Reference Manual [Reference.Digilentinc]", *Reference.Digilentinc.Com*(Webpage, 2019)
<https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual>

Code from 3213 course:

Malagutti, Nicolo, *3213 FPGA Lab Resources* (Australian National University, 2019)

IIT Kampur Electronics Club: Keyboard link

Students.Iitk.Ac.In (Webpage, 2019)
<http://students.iitk.ac.in/eclub/assets/tutorials/keyboard.pdf>

Master constraints File:

"Digilent/Basys3", *Github* (Webpage, 2019)
<https://github.com/Digilent/Basys3/blob/master/Projects/XADC_Demo/src/constraints/Basys3_Master.xdc>

Appendices

Appendix 1: Displayable Characters

A, E, I, O, U, b, C, d, F, G, H, L, n, P, r, S, t,

Note: Mode 3 will also display a flashing | to indicate the cursor position and recognise backspace as a unique input but these are not considered recognised for display purposes. This is the same for the arrow keys which are used in mode 2 and 3.

Appendix 2: Sample sentences

1. COOL CAtS rULE tHE LOnG rED rOAd
2. PASS tHE SALt tO tHE lrOn CHEF
3. tEd IS SAd tO SEE FrEd In bEd
4. tHis IS FOr A HUGE nErd

Appendix 3: SSD decoder segments and recognised characters

- 8'h1C : segments = ~7'b1110111; //A
- 8'h24 : segments = ~7'b1001111; //E
- 8'h43 : segments = ~7'b0110000; //I
- 8'h44 : segments = ~7'b1111110; //O
- 8'h3C : segments = ~7'b0111110; //U
- 8'h32 : segments = ~7'b0011111; //b
- 8'h21 : segments = ~7'b1001110; //C
- 8'h23 : segments = ~7'b0111101; //d
- 8'h2B : segments = ~7'b1000111; //F
- 8'h34 : segments = ~7'b1011111; //G
- 8'h33 : segments = ~7'b0110111; //H
- 8'h4B : segments = ~7'b0001110; //L
- 8'h31 : segments = ~7'b0010101; //n
- 8'h4D : segments = ~7'b1100111; //p
- 8'h2D : segments = ~7'b0000101; //r
- 8'h1B : segments = ~7'b1011011; //S
- 8'h2C : segments = ~7'b0001111; //t
- 8'h5D : segments = ~7'b0000110; //| to display cursor position (not included in recognised characters)
- default : segments = ~7'b0000000;