Exercise 4 Documentation

## Summary of Task:

Exercise 4 should take a string from the terminal emulator, such as putty, which is connected to the SCI1 input port. This string should be stored in a memory location. After the string is stored it should be manipulated such that it is either converted such that all letters following a space are uppercase (and the remaining characters become lowercase), or b. converted so that all letters are uppercase. This decision is done a switch in port p. A copy of the numerical characters will be stored in a new location. The manipulated string is sent to serial with a delay of 1 second. After this has been executed the numerical characters of the string will be sent to the 7 seg which will scroll through these. After which the program will return and wait for a new input.

## Program Summary:

Note detailed documentation for each of these steps can be found in each exercises own documentation\*

* Sets Baud rate and control registers for SCI.
* Jump to receive subroutine for serial input.
  + Wait until TDRE bit is set.
  + Then transmit data into memory location.
  + Once the CR is hit, add newline and CR.
* Jump to Delay subroutine
  + Delay for 1 second
* Load the start of the receive string into x.
  + Store an initial flag into memory locations, this is used to capitalise the first letter of the string if required.
* Jump to the string manipulation subroutine.
  + Check if the string should terminate.
  + Check if char is alpha.
    - Alpha chars get processed.
    - Non alpha chars get skipped.
  + Processed chars have a function applied to them.
    - All upper
    - Upper after a space
* Non valid chars are skipped.
  + Before moving to the next char the current char is tested if it is a space, this information is stored in memory as a flag.
  + Position is incremented and branches back to start of the subroutine.
* Jump to Numerical store Subroutine.
  + Store the numerical characters from the string into a new memory location.
* Jump to transmit subroutine.
  + Wait until TDRF is set then transmit characters.
* Jump to the seven seg display subroutine.
  + Only utilise the scroll function.
  + Add in a feature where if the numerical string is simply the null character, i.e no numerical characters, the scrolling code is skipped and the led remains off.
* Return to the start of the loop

## Discussion Questions:

How can the 7-seg string be displayed if the microprocessor is currently waiting for information from the serial port?

* Initially port b should be cleared such that the LED is disabled.
* At the end of the scroll for the numerical characters it should also be cleared.

What design and planning strategies did you use to incorporate all of the previous exercises into this task?

* For each module there is specified inputs and outputs which would be needed in the pipeline. For example, the string manipulation tasks and the seven seg scrolling functions. Required a string which a way of determining the end of the string.
  + We tried to standardise aspects of the code such as what would be the terminating character.
* We broke the integration task into a flow diagram of what should happen and aimed at decoupling each of the exercises. Whilst this may have not been the most efficient approach, it makes testing much easier as for each stage of the pipeline there are clear inputs and clear desired outputs.
  + This approach is best seen where exercise 1 the string is restored in the memory location and then for exercise 2 this new string is copied to a new memory location for the numerical characters, rather than attempt to do it all in one go.
  + This makes testing easier.
  + Also, where previously in exercise 1 each of the tasks were subroutine functions. We made these branches.
* Other strategies also included reloading registers to ensure that previous operations would not affect the aims, this was particularly important as new sections of the code were appended or added to the old code.

How did you test the code?

* For the testing, please refer to each exercises specific section for testing.
* Exercise 3 for receiving characters testing.
* Exercise 1 for testing the string manipulation.
  + Additionally check using breakpoint that the comparator is working for port H, when 7th bit is 1 it should go to capspace subroutine, if 0 the allupper.
* For the storing of only numerical characters
  + Check that loading #numbers using spc has created free memory.
  + Check numbers and inpstr memory locations have loaded into y and x.
  + Enter all 1 into the terminal, set breakpoints at the numbering subroutine.
  + Check that the asci comparator works as the branch to ‘next’ should not occur.
  + Step through to make sure x is inc and y in inc if there is a number.
  + Check that the subroutine terminates at the carriage char.
* See Exercise 3 for receiving characters testing.
* See Exercise 2 for scrolling through characters

## Diagram:

