Data Compression using Huffman Coding

This is my final project implementation on NXP's FRDM KL25Z board.

Problem Statement:

We often connect our microcontrollers to the computer for debugging purposes. This debug data is sent from the microcontroller to the PC over UART. The transfer of data from a microcontroller to a PC can consume a lot of time. The main reason for this is the size of the huge log messages used for debugging. To speed up the process, we need to reduce the size of this data. This can be done using Huffman coding which enables us to compress data based on the frequency of its ASCII contents.

Objective:

To use Huffman coding for data compression in order to speed up the transfer of messages from FRDM KL25Z to the computer over the serial terminal.

Implementation:

I have implemented Huffman coding for data compression on the FRDM KL25Z board.

Please see the README file for steps to run the application

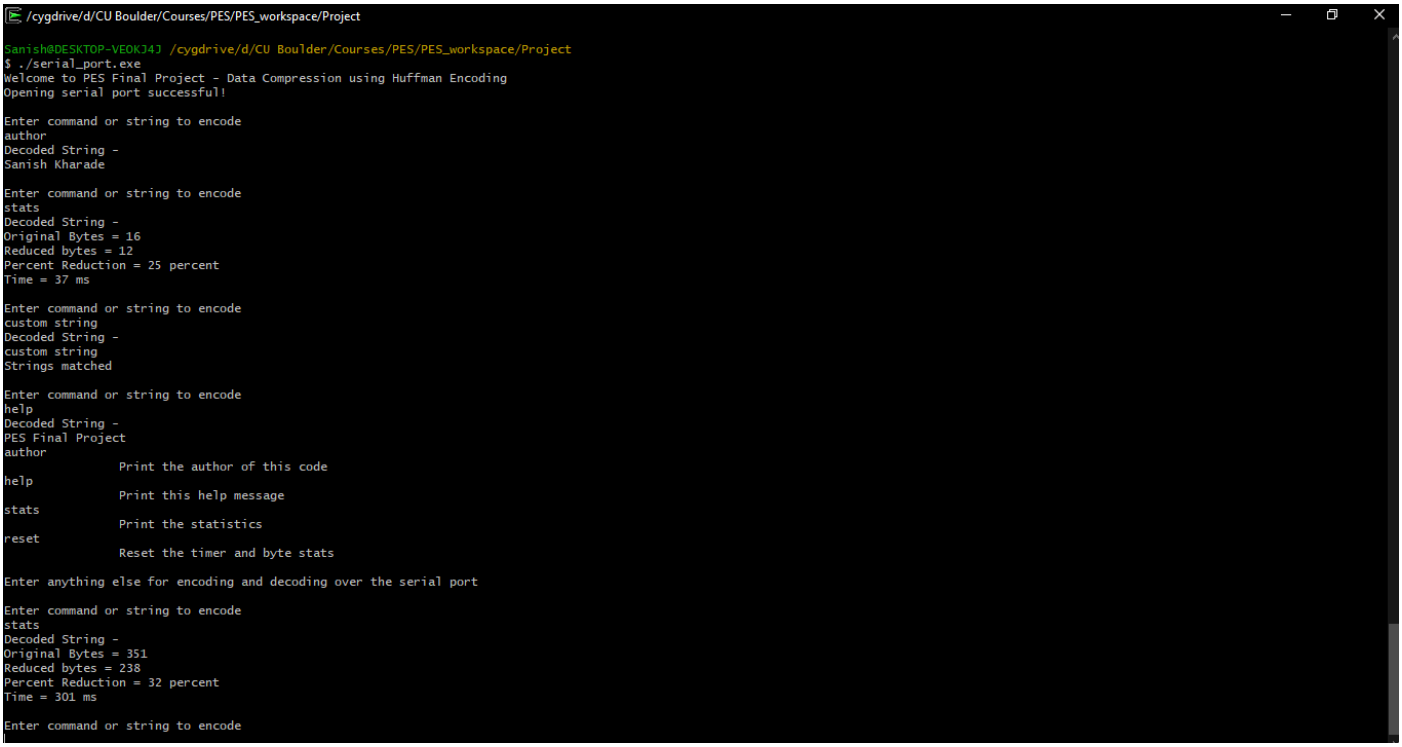
On boot up the microcontroller initializes its clock and all other peripherals. It runs thorough test functions for the circular buffer as well as the Huffman coding algorithm. In the **huffman_test** function, predefined strings are encoded and decoded one after the other. The decoded string is compared with the original string using **strcmp** and **assert** functions. Thus if any test fails, the program will stop running there itself. The testing function is modular and one can easily add more test strings to the two dimensional array without changing anything else in the code. Testing of uppercase, lowercase characters, symbols, special characters, numbers has been done.

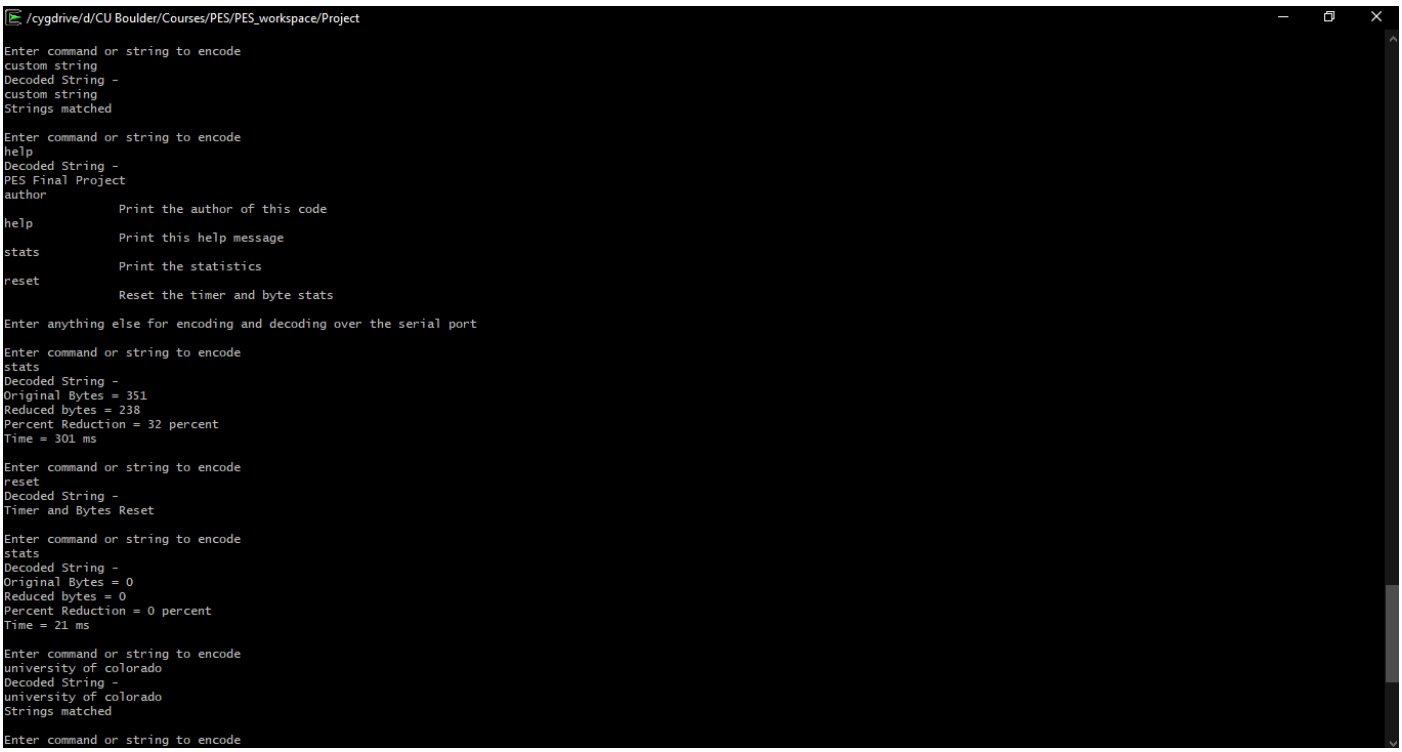
The microcontroller has a command processor running which stores the data received on the serial terminal in a buffer and sends it to command parser once a carriage return is received. I have added 4 commands to this processor. Anything other than the valid commands is considered as a custom string and is encoded on the microcontroller. The encoded string is sent over UART and decoded on the PC end. It is then verified with the original string and printed out.

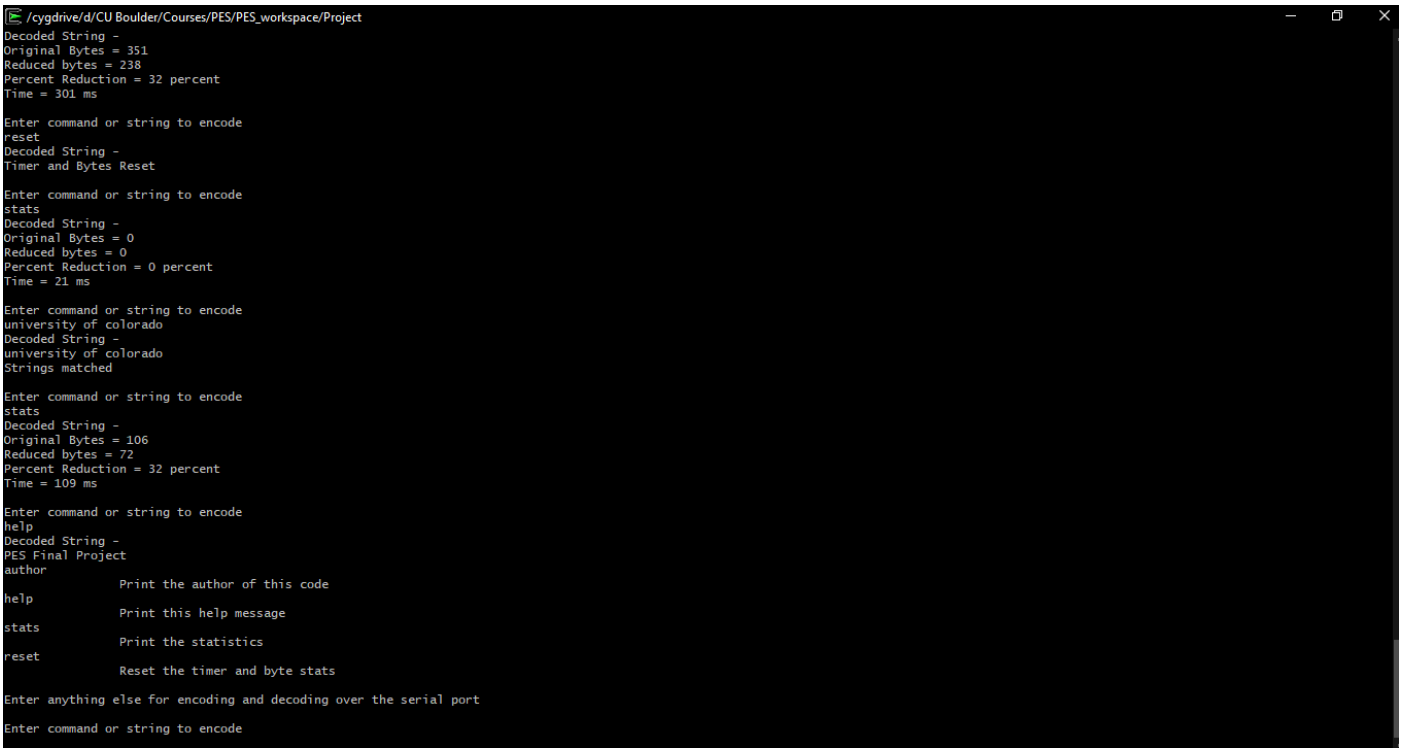
Since the application is meant for reducing size of debug data that flows over the UART, all the data that flows through the serial terminal is encoded using Huffman coding. This is done by passing the buffer received by the **__sys_write** function to the Huffman encoding function. This function will return the amount of data encoded. The decode function on the PC end needs to know how much data has been encoded in order to decode it correctly. Hence this information is first enqueued into the transmit cbfifo. Then the actual encoded data is enqueued on the buffer and sent over UART. Amount of data that flows over the UART is continuously recorded in the **stats** structure which is printed out on receiving the **stats** command. Also, the time taken to transmit the encoded data is also measured using the **systick** timer.

On the PC end the serial connection is opened user input is taken. Then the encoded data sent by the microcontroller is read, decoded and printed out. The characters which have a frequency of zero in the log file will have the code and codebits set to zero in the lookup table. These characters (if entered in the custom string) will not be encoded or decoded and will not appear in the decoded string. Instead the number "6" will appear at the end of the string to indicate this error.

Screenshots:







Conclusion:

Thus I have completed all the milestones mentioned in my proposal.