Data Compression using Huffman Coding

Having worked on NXP's FRDM KL25Z board for over 3 months, it is now time to implement the final course project on the same. Throughout this course I have enhanced my knowledge of timers, circular buffers, state machines, interrupts, serial data transfer, DMA, code optimization etc. I have learnt some of the best coding practices while writing modular and scalable code. It is now time to put all my learnings to work and implement the final project consisting of new topics as well as building on the things learnt so far.

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.

Block Diagram:

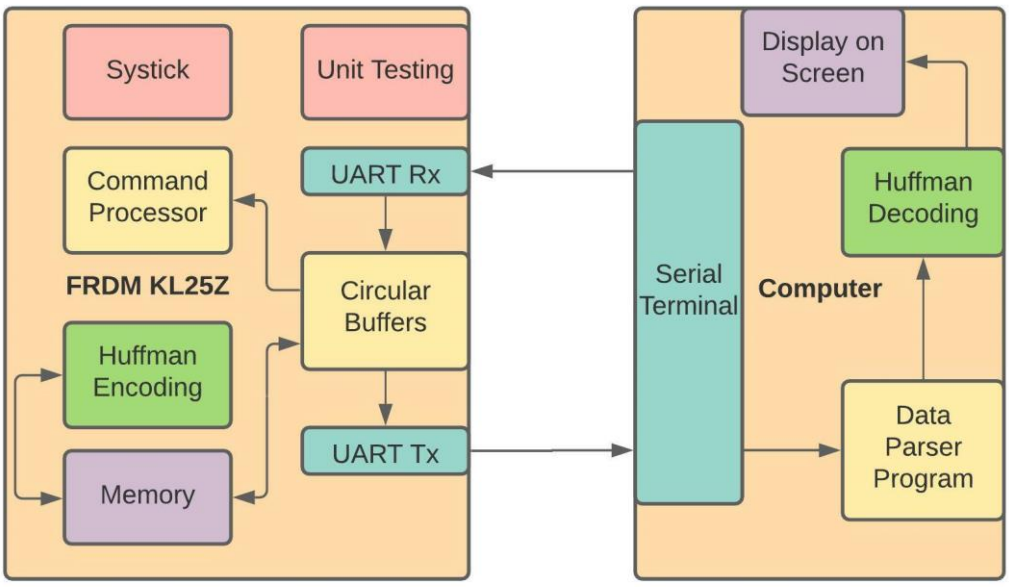

Figure 1: Block Diagram

Implementation:

The system consists of 2 major blocks one on the microcontroller side and one on the PC end. The user will send commands from the PC's serial terminal to the KL25Z via UART. The KL25Z will have a command processor to parse these commands, interpret them and perform the correct function. This command could either be a compression test which will compress a predefined string stored on the KL25Z and send it over UART; or it could be a variable string compression in which the user can manually send a string to the microcontroller for compression. In both cases the original string and the compressed (encoded) string are sent back to the PC where a different program will be used to decompress the encoded data. The decompressed data will be compared to the original data to check for any losses. I will be using Huffman coding for the compression and decompression process. I will use the systick timer to keep track of time required for the data transfer of both the original and compressed data. The time and size reduction statistics will be reported to the user and will vary for different data sets. I will have a debug build target in which I will thoroughly test the Huffman coding and decoding along with testing the circular buffers.

Stretch Goal – If I am able to achieve the above mentioned functionality well within the assigned time, I plan to learn about unit testing using uCunit / Cmocka and implement it in my test suite.

Key Milestones:

I have divided the project into multiple sub parts and will be following the below mentioned sequence-

- First, I will write the program for Huffman encoding and decoding independent of the microcontroller. I will use automated tests to test it thoroughly using multiple strings with different frequencies of ASCII data.
- Once the Huffman coding is working, I will write the program on the PC end to take control of the UART. This program will be able to send and receive data to the microcontroller via the serial terminal.
- Then I will work on the UART program on the MCU side to send the original and compressed data to the PC.
- Lastly I will write the command processor which gives the user multiple options to choose from.
- Finally, I will try to work on the stretch goal of using testing frameworks like uCunit / Cmocka.

FAQs:

1. What functionality will your project demonstrate?

The user will send some data from the UART to the KL25Z, which will be compressed and sent back via UART to the PC where it will get decompressed. The data compression will reduce space usage and will speed up the data transfer over UART. The program will also monitor the percentage reduction in size and the time taken for the transfer. These details will be displayed to the user on the terminal.

2. What technologies will you use? For those areas we have covered in class, how will you demonstrate deeper knowledge than what we covered in the biweekly homework assignments?

Technologies that I will be using in my project are -

- UART

- Timers
- Interrupts
- Circular Buffers
- Data Compression
- Command Processing
- Unit Testing using uCunit/Cmocka (Stretch goal)

The project might involve tweaking the default parameters for peripherals which will require me to go through the data sheet and reference manual.

3. What do you anticipate needing to learn in order to develop your project? What sources (KL25Z Reference Manual, Internet sites, etc.) do you plan to use to figure out how to do whatever it is you are attempting?

The new things that I need to learn in order to complete my project are -

- I will need to thoroughly understand how Huffman encoding and decoding works and how the binary tree for the Huffman code is formed depending on the frequencies of ASCII characters.
- I will also have to learn how to write a C/Python program to read and send data over UART from a Windows PC

I will be using the following references for the project

- FRDM KL25Z reference manual, datasheet, user guide.
- https://en.wikipedia.org/wiki/Huffman_coding
- https://aticleworld.com/serial-port-programming-using-win32-api/
- https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

4. Does your project require any additional hardware? If so, what will you acquire, and what is your plan for assembly?

No, the project does not require any additional hardware.

5. Finally, what is your testing strategy for your project? Will you develop automated tests, will you use manual tests, or will you use a mixture of both?

I will use automated tests by using “asserts”. This will be done in the debug build. I will thoroughly test Huffman coding and the circular buffers used for serial transfer of data. Since Huffman coding depends on the frequency of its ASCII character contents, multiple data sets with same size might undergo different levels of compression.

Example - “AAAAAAAABBBBBBBB” and “ABCDEFGHABABAACD” will have different compressed sizes even though they have the same size originally. I will ensure that I test all such cases.

The command processor will deal with handling invalid user inputs.

Stretch Goal – I will use one of the testing frameworks (uCunit / Cmocka) for testing.

Conclusion:

I am looking forward to working on this project, learning new stuff and overcoming all challenges I face along the way.