Read Me

Perla Vanessa Jaime Gaytán A00344428
Edgar Damián Reyes Herrera A01634031

October 24th, 2020

1 Introduction

For this program we used an FRDM KL25Z Micro-controller, so we didn't have to make use of QEMU.
The main challenge we faced was to make the UART protocol work based on interrupts as opposed to polling. Once the UART protocol was up and running, we were tasked with echoing inputs in to the serial communications port, except that the program not only must echo the inputs, it must invert lower case characters to upper case and vice versa. Also, the main program must end if it receives the string "end".

Figure 1: FRDM kl25z

2 Program Flow

Our tool-chain is very simple since we used the Code Warrior compiler that generates all the necessary material for our board KL25 to interpreted our code. This includes an assembly program, a makefile, an elf file and many more auto-generated files that made it possible to successfully test our program.
After variable declaration we must initialize the UART protocol, which involves connecting the clock to port A, configuring two ports A as receiver and transmitter and setting it to output. When the UART is running, the blue led is

toogling, but when the word end is entered (could be lower case, upper case or any variation) the led turns off and the uart is disable.

The way that interruptions are managed is that we check every character that is printed and execute an interruption per character, if our characters are lower case we subtract 32 bits from the ASCII code and make it upper case and then print it (the same is done in reverse for upper case letters), if said characters happens to spell out "end", the program will detect it and close the main ending the program. It also disable the uart when the "end" word is entered.
The specific program flow is made so that the main is always running and the interrupt handler is working intermittently alongside the UART transmitter/receiver.
It's also worth to mention that we didn't need the code segments related to "lock and unlock" since our code is designed for a physical card and the interrupt handler was coded as such.

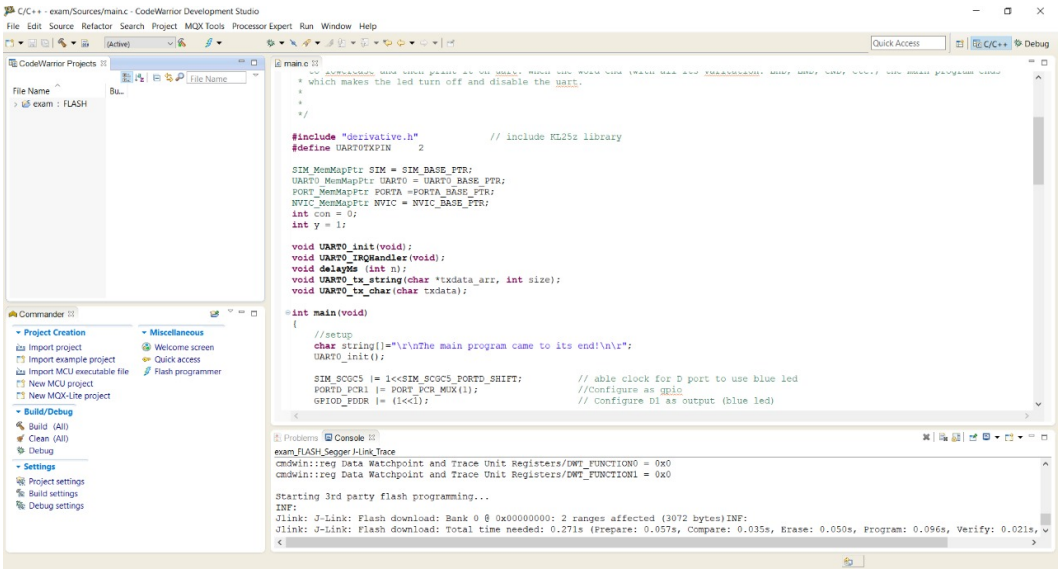

Figure 2: A screenshot of our code and the compiler running it

3 Conclusion

Despite being a very basic communication protocol, UART can come in handy when dealing with higher level projects, it's also advantageous that its a well documented protocol, which makes for a good starting point when trying to learn serial communication.

Link to the video: `https://youtu.be/pU3x_Foi0d8`