Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/27/2021

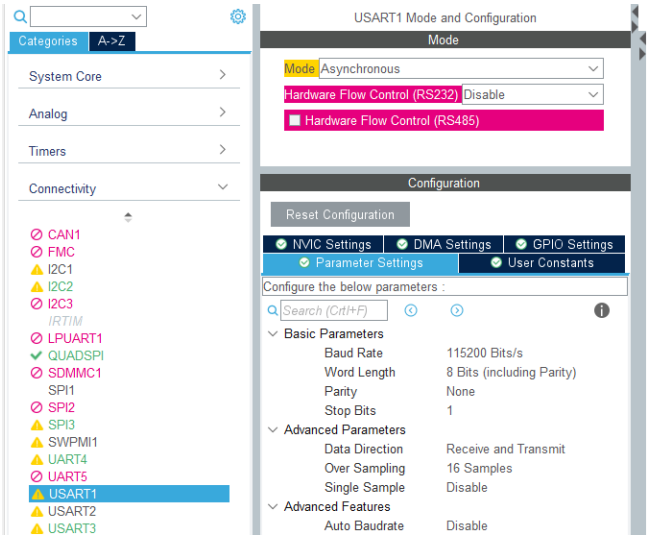
Assignment 7: UART

The following will document completion of the seventh assignment for ECE-40291, with the stated goals of:

1. Your skills for using the HAL Generic UART driver to send messages out the ST-LINK Virtual COMP Port, or UART1, and display the messages on your host computer. Display the message "UART1: 0, UART1: 1, ...UART1: n", where N is a count that increments every 5 seconds.
2. Your skills for using the HAL Generic UART driver to send messages out the Arduino TX/RX Port, or UART4, and display the messages on your host computer. Display the message "UART4: 0, UART4: 1, ...UART4: n", where N is a count that increments every 10 seconds.
3. Use a program such as PuTTY (PC) or minicom (Linux) or screen (Mac) to display the output on the virtual serial console.

1. Your skills for using the HAL Generic UART driver to send messages out the ST-LINK Virtual COMP Port, or UART1, and display the messages on your host computer. Display the message "UART1: 0, UART1: 1, ...UART1: n", where N is a count that increments every 5 seconds.

We will begin, as always, by generating the default project files for our Disco board within the IDE. Once the base files have been generated, we can enter the configurator and ensure that USART1 is using to our desired (default) settings, 115.2k, 8, N, 1.

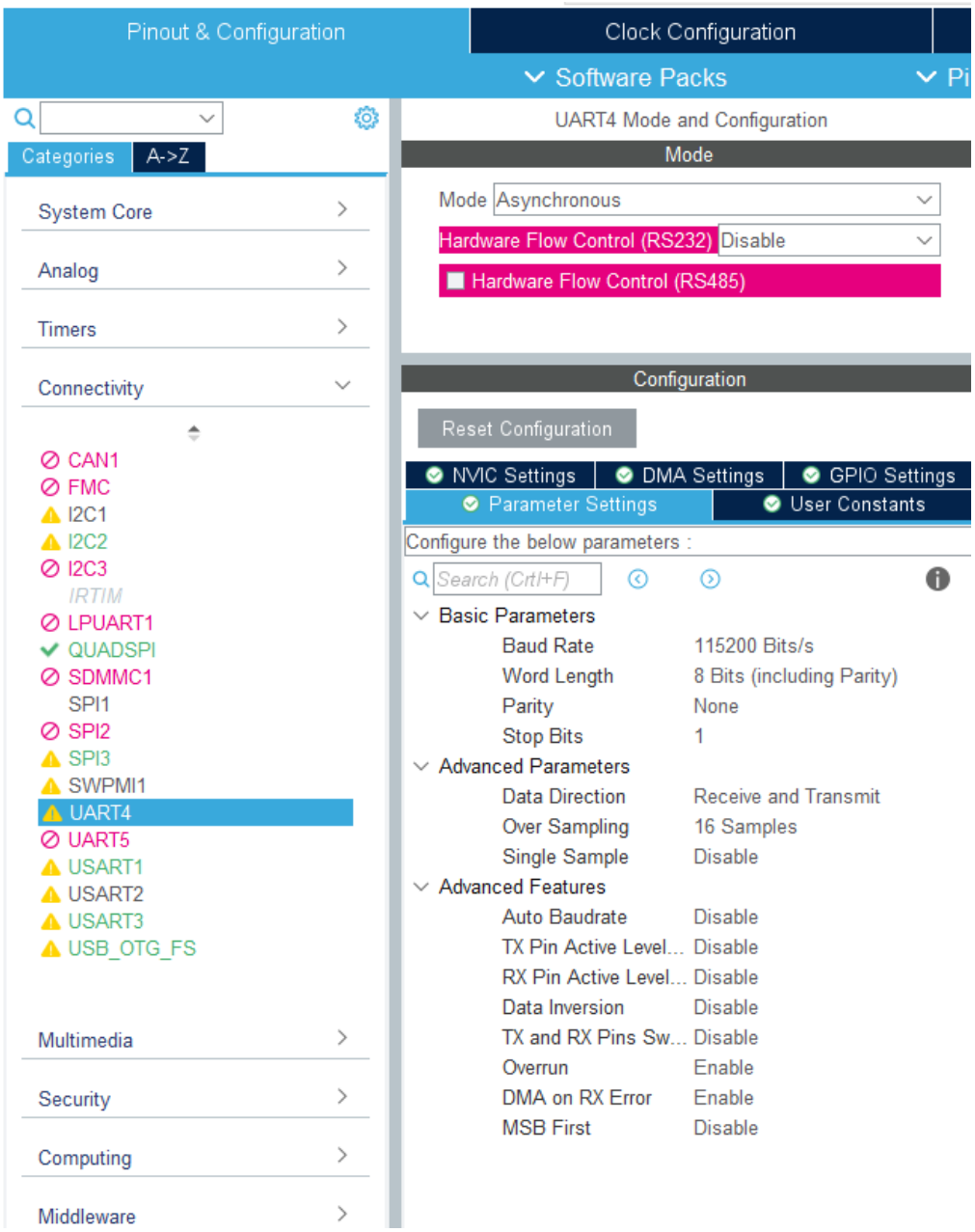


Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/27/2021

2. **Your skills for using the HAL Generic UART driver to send messages out the Arduino TX/RX Port, or UART4, and display the messages on your host computer. Display the message "UART4: 0, UART4: 1, ...UART4: n", where N is a count that increments every 10 seconds.**

Moving to UART4, we will enable it and confirm the same settings. Closing the configuration interface should rebuild and apply all changes necessary to the project files.



Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/27/2021

Finally, we can move in to *main.c* and develop a simple counter/loop structure that will be used to print on the appropriate serial ports at the prescribed times, every 5 seconds for USART1 and every 10 for UART4.

```
121   /* USER CODE BEGIN 2 */
122
123   char buffer[100] = {0};
124   uint8_t uart1Counter = 0;
125   uint8_t uart4Counter = 0;
126
127   snprintf(buffer, sizeof(buffer), "UART1: %d\n", uart1Counter);
128   HAL_UART_Transmit(&huart1, (uint8_t*) buffer, strlen(buffer), 1000);
129
130   snprintf(buffer, sizeof(buffer), "UART4: %d\n", uart4Counter);
131   HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
132
133   /* USER CODE END 2 */
```

This first block of code serves to set up our buffer and counter declarations, which will be used to compare to another counter to determine when to print later, as well as the initial 0-count print statements for both serial ports.

```
137   while (1)
138   {
139     /* USER CODE END WHILE */
140     /* USER CODE BEGIN 3 */
141
142       for (uint16_t loopCounter = 1; loopCounter < 1000; loopCounter++)
143         {
144           HAL_Delay(1000);
145
146           if ((loopCounter % 5) == 0)
147           {
148               uart1Counter++;
149               snprintf(buffer, sizeof(buffer), "UART1: %d\n", uart1Counter);
150               HAL_UART_Transmit(&huart1, (uint8_t*) buffer, strlen(buffer), 1000);
151           }
152           if ((loopCounter % 10) == 0)
153           {
154               uart4Counter++;
155               snprintf(buffer, sizeof(buffer), "UART4: %d\n", uart4Counter);
156               HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
157           }
158         }
159   }
160   /* USER CODE END 3 */
```

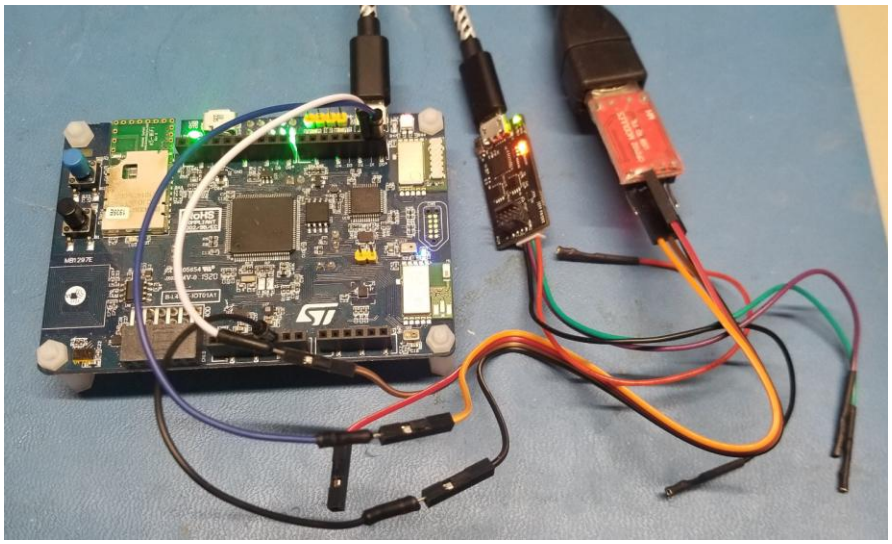
The second block, in the *while(1)* loop, will throw us into a loop counting up to an arbitrarily large value (1000 here), something big enough that we don't roll over too soon, and after a 1 second delay, compare the accrued value via modulo division to our assigned delays of 5 and 10 seconds, printing if either of them are equal to 0.

Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/27/2021

3. Use a program such as PuTTY (PC) or minicom (Linux) or screen (Mac) to display the output on the virtual serial console.

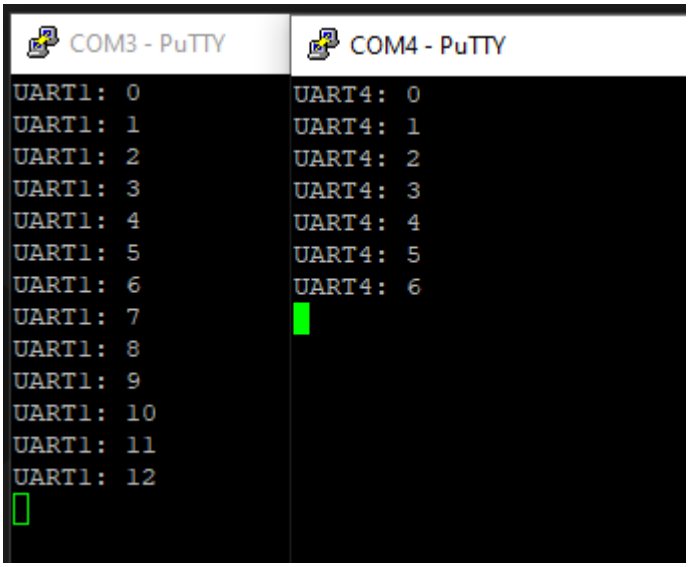
With the code developed and successfully compiled, we can flash the board and hook up our UART4 hardware. There was some back and forth on the discussion board over whether this should be accomplished with a loopback method or a separate piece of communication hardware. Having already accomplished the loopback method in the ECE-40293 UART assignment, I elected for the extra hardware route. This had the end result of making the above code simpler, not having to deal with defining interrupt callbacks, and making a bit of a rats nest on my work bench. I also discovered that for some reason on my Windows machine, my Black Magic probe is recognized and instantiates as two COM ports but neither of them worked for data RX. Switching over to an SI Labs CP2102 dongle from my toolbox, I began to receive data immediately. That's an issue for another time but it was interesting as part of what makes the BMP so handy is the combo SWD/UART in one piece of hardware and I've never had issues on any of my Linux machines. We can see in the below image the mess of wires tying the board's GNDs together and the dongle's TX pin to the Arduino D0/ RX header (unnecessary in this example) and the dongle's RX pin to the Arduino D1/TX header.



Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/27/2021

With everything properly connected, open an instance of PuTTY for each COM port, reset the board to restart the counts, and we see USART1 incrementing and printing at 2x the frequency of UART4.



Closing Thoughts

Much like the ADC lesson, this was very easy to accomplish using the skills developed under the ECE-40293 UART assignment. Other than serving as reinforcement to that learning, I think the best part of this assignment was the large amount of discourse it prompted from students on the discussion board. I enjoyed the opportunity to review their examples and suggestions and appreciated the chance to compare my style and methods to others in the group.