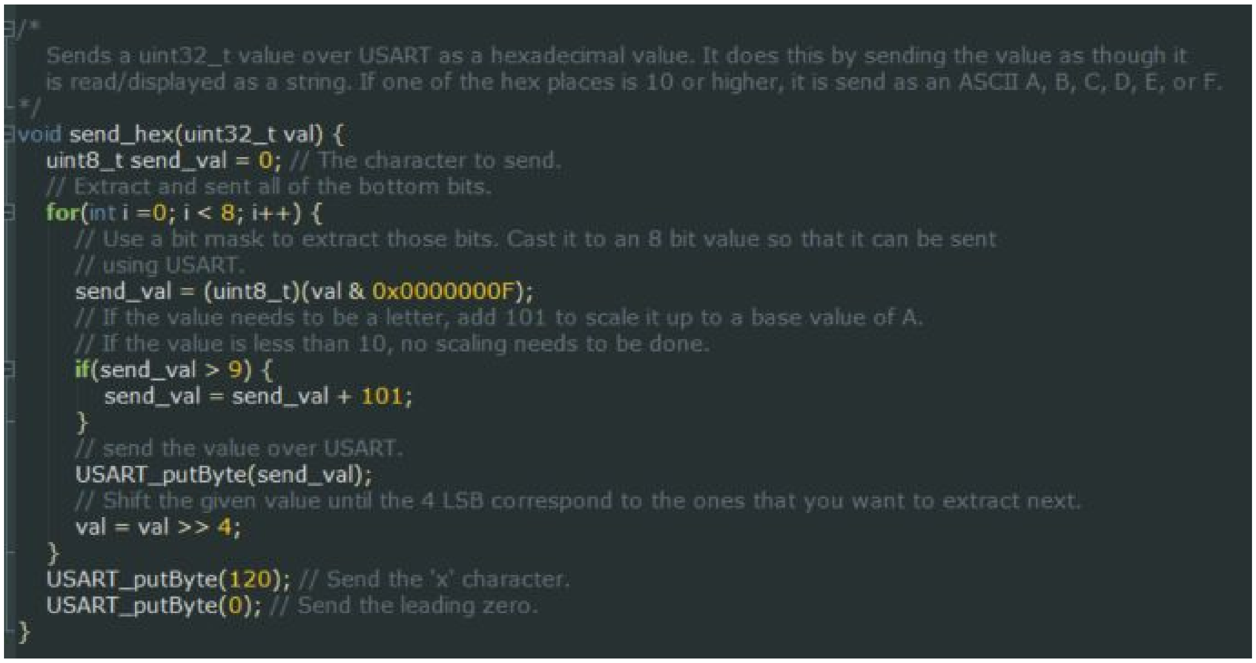
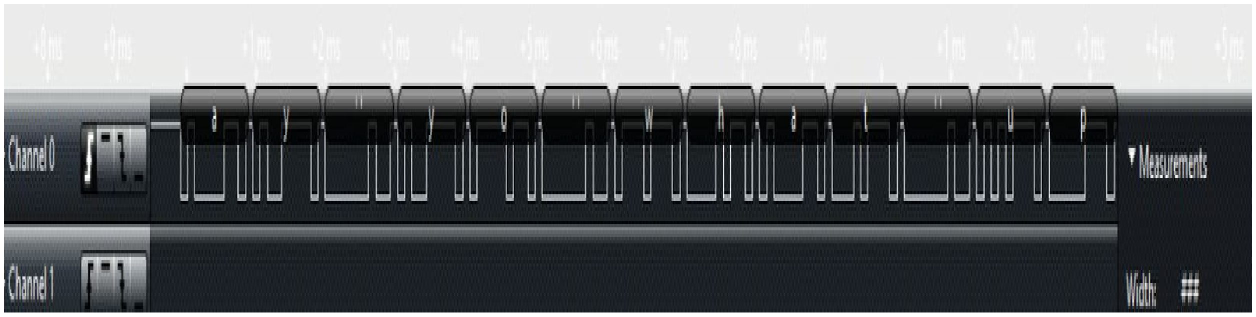
Dusty Argyle
Max Hansen
ECE 5780
Lab 4 (USART)

1. The delay_ms() function does what the name describes; it delays for a set amount of time. It does this by first setting a counter value (in this case 125). It then enters a while loop until this counter is equal to 0. During this time, the processor will be put to sleep. Then, every time the Systick interrupt occurs, this counter will decrement. In this case it's once every millisecond. So while the counter is non-zero, then the processor is put to sleep unless an interrupt occurs.
2. Putting the processor to sleep doesn't cause an issue to the rest of the program because the processor isn't supposed to be doing anything. When the counter goes to zero, an interrupt occurs which wakes the processor back up, and does not run into the wfi command until the next delay.
3. When the button is pressed and its interrupt is enabled, then it turns off the interrupt. When the Systick timer interrupt goes off, then it turns the external interrupt back on using the callback. This is a simple debounce because there will be a short amount of time (1ms at most) until the external interrupt will be able to fire again, which means that any bouncing effects will be ignored.
4. The NULL character at the end of the string is important because the while loop will continue until the value at **string* is either zero or NULL (which is the ASCII character value of 0). The double quotes will automatically tack on the NULL character, so the while loop will send all of the characters until it reaches the NULL character. This means that all of the characters have been sent and to end the transmission through the USART.
5. One of the features of the USART is that it has direct memory access (DMA). The registers that control this are in USART_TDR and USART_CR3. Another feature is that it has a single wire, half duplex mode.This is set/controlled by bits in USART_CR2 (bits LINEN and CLKEN) and USART_CR3 (bits SCEN, IREN, and HDSEL). Another, third feature that it has is multiprocessor communication. This is controlled using registers USART_CR2 (bit LINDEN) and USART_CR3 (bits HDSEL, IREN, and SCEN). A last feature (that will be mentioned) is the auto baud rate detection. This is controlled using registers USART_CR2 and setting the bit ABREN.
6. Here is a copy and pasted text version, as well as a screenshot of it in a text editor, in case that's easier to read.

```
/*
        Sends a uint32_t value over USART as a hexadecimal value. It does this by sending the
        value as though it is read/displayed as a string. If one of the hex places is 10 or higher, it
is
        send as an ASCII A, B, C, D, E, or F.
*/
void send_hex(uint32_t val) {
        uint8_t send_val = 0; // The character to send.
        // Extract and sent all of the bottom bits.
        for(int i =0; i < 8; i++) {
                // Use a bit mask to extract those bits. Cast it to an 8 bit value so that it can be
sent
                // using USART.
                send_val = (uint8_t)(val & 0x0000000F);
                // If the value needs to be a letter, add 101 to scale it up to a base value of A.
                // If the value is less than 10, no scaling needs to be done.
                if(send_val > 9) {
                        send_val = send_val + 101;
                }
                // send the value over USART.
                USART_putByte(send_val);
                // Shift the given value until the 4 LSB correspond to the ones that you want to
                // extract next.
                val = val >> 4;
        }
        USART_putByte(120); // Send the 'x' character.
        USART_putByte(0); // Send the leading zero.
}
```



```
/*
   Sends a uint32_t value over USART as a hexadecimal value. It does this by sending the value as though it
   is read/displayed as a string. If one of the hex places is 10 or higher, it is send as an ASCII A, B, C, D, E, or F.
*/
void send_hex(uint32_t val) {
   uint8_t send_val = 0; // The character to send.
   // Extract and sent all of the bottom bits.
   for(int i =0; i < 8; i++) {
      // Use a bit mask to extract those bits. Cast it to an 8 bit value so that it can be sent
      // using USART.
      send_val = (uint8_t)(val & 0x0000000F);
      // If the value needs to be a letter, add 101 to scale it up to a base value of A.
      // If the value is less than 10, no scaling needs to be done.
      if(send_val > 9) {
         send_val = send_val + 101;
      }
      // send the value over USART.
      USART_putByte(send_val);
      // Shift the given value until the 4 LSB correspond to the ones that you want to extract next.
      val = val >> 4;
   }
   USART_putByte(120); // Send the 'x' character.
   USART_putByte(0); // Send the leading zero.
}
```



Screenshot of logic analyzer input when sending a string over USART. Didn't copy to document very well, so it is also attached to the lab submission.