ECE4144 – Communication Hands On Assignment

The objective of this hands-on assignment is to implement a simple communication program based on USART built on the board. The processor has only one USART, which is UART1 (instead of what written in the document). Based on the datasheet, I have implemented the following code to fulfill requirements:

```
/* UCSR1A is set as default.
Normal transmission speed,
disable the multi-processor communication mode */
UCSR1A = 0x00;            // Reset the UCSR1A

UCSR1B = 0x00;            // Reset the UCSR1B
UCSR1B |= (1 << RXCIE1) | // Enable RX Complete Interrupt
        (1 << RXEN1)  |   // Enable Receiver
        (1 << TXEN1);     // Enable Transmitter

UCSR1C = 0x00;            // Reset the UCSR1C
UCSR1C |= (1 << UCSZ11) | // Set Character Size to 8-bit
        (1 << UCSZ10);

UBRR1 = 51;               // UBRR1 = (fosc / (16 * Baud Rate)) - 1
                          //       = (8MHz / (16 * 9600)) - 1 = 51.08
```

After following the requirements in question 2 and question 3, we can have the following code:

```
#include <Arduino.h>

char receivedByte = 0;

void USART_Init() {
  /* UCSR1A is set as default.
     Normal transmission speed,
     disable the multi-processor communication mode */
  UCSR1A = 0x00;            // Reset the UCSR1A

  UCSR1B = 0x00;            // Reset the UCSR1B
  UCSR1B |= (1 << RXCIE1) | // Enable RX Complete Interrupt
            (1 << RXEN1)  | // Enable Receiver
            (1 << TXEN1);   // Enable Transmitter

  UCSR1C = 0x00;            // Reset the UCSR1C
  UCSR1C |= (1 << UCSZ11) | // Set Character Size to 8-bit
            (1 << UCSZ10);

  UBRR1 = 51;               // UBRR1 = (fosc / (16 * Baud Rate)) - 1
                            //       = (8MHz / (16 * 9600)) - 1 = 51.08
}

ISR(USART1_RX_vect) {
  receivedByte = UDR1;      // Read the received byte

}

void TransmitString(const char* str, uint8_t length) {
  // Transmit byte by byte
  for (uint8_t i = 0; i < length; i++) {
    while (!(UCSR1A & (1 << UDRE1))) {
      // Wait for the transmit buffer to be empty
    }
    UDR1 = str[i]; // Transmit the byte
  }
}

char GetNextReceivedByte(){
  char byte;
  cli();               // Disable global interrupts
  byte = receivedByte; // Read the received byte
  receivedByte = 0;    // Clear the received byte after reading
  sei();               // Enable global interrupts
  return byte;
}

void setup() {
  USART_Init(); // Initialize USART1
  sei();        // Enable global interrupts
}

void loop() {
  char currentByte = GetNextReceivedByte();
  // Check if a byte is received
  if (currentByte != 0) {
    switch (currentByte) {
      case '1':
        TransmitString("One\n", 4);
        break;
      case '2':
        TransmitString("Two\n", 4);
        break;
      default:
        TransmitString("Default\n", 8);
        break;
    }
  }
}
```

After uploaded the program to the board, I connected the board to the FTDI programmer like in the figure below:

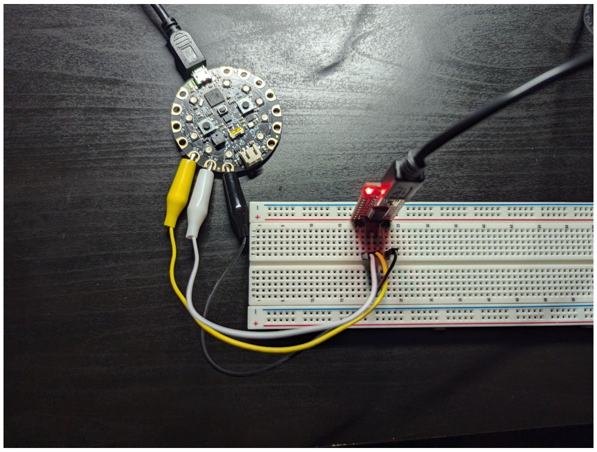
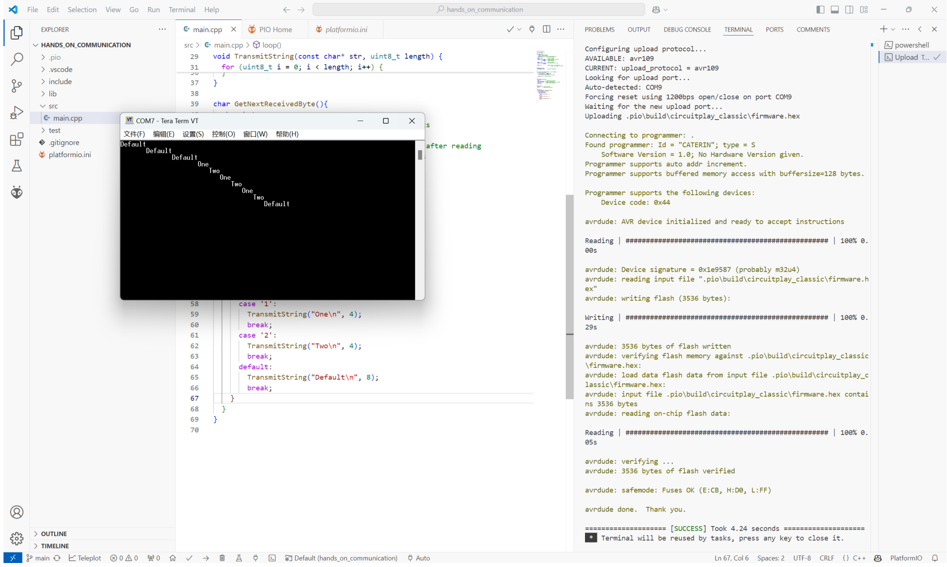


Figure 1: Connection between the board and the FTDI programmer

Then, I opened the serial monitor in the Tera Term and typed along the commands. Then, I received the outputs like in the figure below:


File Edit Selection View Go Run Terminal Help
hands_on_communication
EXPLORER
HANDS_ON_COMMUNICATION
.pio
.vscode
include
lib
src
main.cpp
test
.gitignore
platformio.ini
OUTLINE
TIMELINE
main.cpp
PIO Home
platformio.ini
src > main.cpp > loop()
COM7 - Tera Term VT
文件(F) 编辑(E) 设置(S) 控制(O) 窗口(W) 帮助(H)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
powershell
Upload T...
main Teleplot 0 0 0 Default (hands_on_communication) Auto
Ln 67, Col 6 Spaces: 2 UTF-8 CRLF {} C++ PlatformIO

```
void TransmitString(const char* str, uint8_t length) {
  for (uint8_t i = 0; i < length; i++) {
}

char GetNextReceivedByte(){
```

```
Default
       Default
              Default
                     One
                        Two
                           One
                              Two
                                 One
                                    Two
                                       Default
```

```
    case '1':
      TransmitString("One\n", 4);
      break;
    case '2':
      TransmitString("Two\n", 4);
      break;
    default:
      TransmitString("Default\n", 8);
      break;
  }
 }
}
```

```
Configuring upload protocol...
AVAILABLE: avr109
CURRENT: upload_protocol = avr109
Looking for upload port...
Auto-detected: COM9
Forcing reset using 1200bps open/close on port COM9
Waiting for the new upload port...
Uploading .pio\build\circuitplay_classic\firmware.hex

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
    Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
    Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.
00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading input file ".pio\build\circuitplay_classic\firmware.h
ex"
avrdude: writing flash (3536 bytes):

Writing | ################################################## | 100% 0.
29s

avrdude: 3536 bytes of flash written
avrdude: verifying flash memory against .pio\build\circuitplay_classic
\firmware.hex:
avrdude: load data flash data from input file .pio\build\circuitplay_c
lassic\firmware.hex:
avrdude: input file .pio\build\circuitplay_classic\firmware.hex contai
ns 3536 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.
05s

avrdude: verifying ...
avrdude: 3536 bytes of flash verified

avrdude: safemode: Fuses OK (E:CB, H:D0, L:FF)

avrdude done.  Thank you.

==================== [SUCCESS] Took 4.24 seconds ====================
 *  Terminal will be reused by tasks, press any key to close it.
```

Figure 2: Output of the program

which is as expected.