**2020F_ESE 3014_1**

**SEMESTER:** 3rd

**INSTRUCTOR:** Linchen Wang

**LAB:** 8

**DUE DATE:** 29 Nov 2020 - 23:55

**SUBMITTED DATE:** 29 NOV 2020

**STUDENTS:** Amandeep Singh (C0765434)

**Lab-8**

## INTRODUCTION

In this Lab we are going to use the FTDI cable to communicate between the beaglebone and the host machine using UART.

## DESCRIPTION

- Pin Configuration:
  We can use the following schematics to understand how the connections are supposed to be made.

| | | | |
|---|---|---|---|
| DGND | 1 | 2 | DGND |
| VDD_3V3 | 3 | 4 | VDD_3V3 |
| VDD_5V | 5 | 6 | VDD_5V |
| SYS_5V | 7 | 8 | SYS_5V |
| PWR_BUT | 9 | 10 | SYS_RESETN |
| UART4_RXD | 11 | 12 | GPIO_60 |
| UART4_TXD | 13 | 14 | GPIO_40 |
| GPIO_48 | 15 | 16 | GPIO_51 |
| GPIO_4 | 17 | 18 | GPIO_5 |
| UART1_RTSN | 19 | 20 | UART1_CTSN |
| UART2_TXD | 21 | 22 | UART2_RXD |
| GPIO_49 | 23 | 24 | UART1_TXD |
| GPIO_117 | 25 | 26 | UART1_RXD |
| GPIO_125 | 27 | 28 | GPIO_123 |
| GPIO_121 | 29 | 30 | GPIO_122 |
| GPIO_120 | 31 | 32 | VDD_ADC |
| AIN4 | 33 | 34 | GNDA_ADC |
| AIN6 | 35 | 36 | AIN5 |
| AIN2 | 37 | 38 | AIN3 |
| AIN0 | 39 | 40 | AIN1 |
| GPIO_20 | 41 | 42 | UART3_TXD |
| DGND | 43 | 44 | DGND |
| DGND | 45 | 46 | DGND |

1. Once the connection are made we can start with the command “dmesg|grep tty” to get the name of the FTDI cable connected to the host machine.

   ```
   $ dmesg|grep tty
   ```

   Which would result in something like:

   ```
   [ 0.001982] console [tty0] enabled
   [ 686.529224] usb 2-2: pl2303 converter now attached to ttyUSB0
   ```

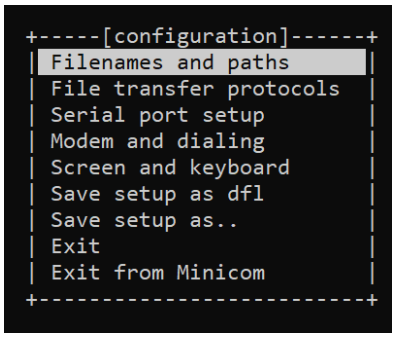
   Here, ‘ttyUSB0’ is out FTDI cable.

2. So now we can use this information to open up the minicom window for the following ‘ttyusb0’, this can be done in two ways:

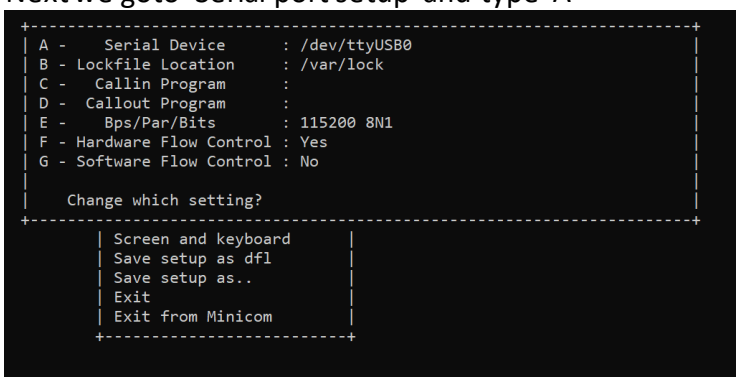
   ```
   $minicom -b 115200 -o -D /dev/ttyUSB0
   ```
   Where 115200 is the baud rate

Or we can manually select ttyUSB0 under the minicom window, as shown below:

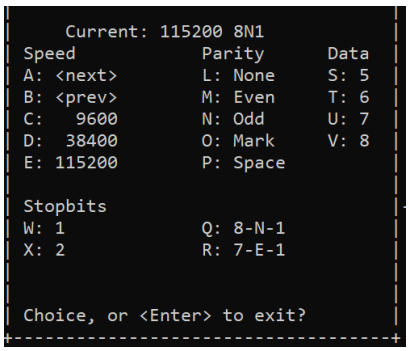
- `$minicom -s`




- Next we goto 'Serial port setup' and type 'A'




Note: here we could have also changed the baut rate according to our uart.c file by type 'E'



**3.** Now we get to the coding part:

```
if ((file = open("/dev/ttyO4", O_RDWR | O_NOCTTY | O_NDELAY))<0)
   {
      perror("UART: Failed to open the device.\n");
      return -1;
   }

struct termios options;

tcgetattr(file, &options);
options.c_cflag = B115200 | CS8 | CREAD | CLOCAL;
options.c_iflag = IGNPAR;
tcflush(file, TCIFLUSH);
tcsetattr(file, TCSANOW, &options);
```

- This code snippet covers most of the code, here we open the tty port that we are going to be using and check if the port is opened or not.
- Next we set an object to the '`struct termios`' that we are going to use to set a few important flags to be able to take input and output from the minicom windows
- We start with '`tcgetattr()`' to get the attributes of the 'file' and the struct, next we set the baud rate and the modes in which we want the flag to work in. Once the baud rate is set the input flag or '`options.c_iflag`' can be set to `IGNPAR` which basically means ignore framing and parity errors. Finally we use '`tcsetattr`' to set the attributes

**Following the steps:**

1. boot your Linux host machine
2. insert the FTDI cable into your host machine's USB port

**3.** type dmesg | tail, and interpret what you are seeing (**please refer to POINT 1**)

**a.** a new device should have been created; what is it called?

ttyUSB0

**b.** what kind of device is it?

USB2

**c.** are there any performance limitations on the USB device, now that at VCP driver is in place?

The length of the cable affects the transfers speeds.

**6**. using Derek Molloy's uart.c code as a starting point, can you transmit some sample strings to your host machine? (**please refer to the appendix**)

**a.** what is the maximum data rate at which you can transmit data over this USB-VCP channel?

115200 bit per second

**b.** what protocol are you using?

Modem

**CONCLUSION**

- Lab-8 was really important to understand minicom and how baud rate can affect the data transfer times.
- The FTDI cable has way better speed as compared to normal data transfer over wifi.

**APPENDIX**

- Youtube link: https://www.youtube.com/watch?v=FR1K_z3nMKw
- new.c header files:

```
#include <stdio.h>
#include <termios.h>
#include <fcntl.h>
#include <unistd.h>
```

- new.c Code:

```
int main()
{
int file, count;

if ((file = open("/dev/ttyOX", O_RDWR | O_NOCTTY | O_NDELAY))<0)
{
      perror("UART: Failed to open the device.\n");
      return -1;
}
 struct termios options;
   tcgetattr(file, &options);
   options.c_cflag = B115200 | CS8 | CREAD | CLOCAL;
   options.c_iflag = IGNPAR;
   tcflush(file, TCIFLUSH);
   tcsetattr(file, TCSANOW, &options);

unsigned char tx[18]= "hello Beaglebone!";

if ((count = write(file,&tx,18))<0)       //writes to the file
{
perror("failed to write to the output");
return -1;
}

unsigned char rx[100];

if ((count = read(file,(void*)rx,100))<0) //reads from the file
{
perror("failed to read from the output");
return -1;
}

if (count==0)
       printf("there was no data to read!");
close(file);
return 0;
}
```