U A R T

C O M M U N I C A T I O N

b/w

raspberry pi [version 1, model B]

and linux terminal [FTDI]

presented by-

SHARAN KUMAR

RAVI KUMAR KUSHAWAHA

SAQIB AZIM

Getting started with UART communication:

- ❖ Raspberry pi gpio pins 8 and 10 –are respectively RX and TX.
- ❖ RS-232 pins 1 and 2 -are respectively RX and TX
- ❖ Ground of both are connected.
- ❖ Using g++ compiler to compile c program.
- ❖ Using termios header file for uart.
- ❖ Sending and recieving of data on terminal.

What's next:

- In the code including appropriate header files.
- Opening serial port
- Setting up baud rate, character size, parity bits (optional)
- Not enabling N_DELAY (i.e. By default blocking mode is on, input buffer will wait till it receives data)
- Enable -> sudo chmod a+rw /dev/ttyAMA0 to give permission to open uart port
- For LINUX replace AMA0 with USB0

Create a file say uart.c or uart.cpp

Open the file and start writing

Writing the code:

Including header files

```
#include <stdio.h>

Following header files are used for uart

#include <unistd.h>           //  (unix std library- for POSUX functions)
#include <fcntl.h>            //  (function control)
#include <termios.h>         //   (terminal i/o)
```

Introducing a file descriptor to open serial port else return -1 in case of error

```
int main(){
      int uart0_filestream = -1;


      uart0_filestream = open("/dev/ttyUSB0", O_RDWR | O_NOCTTY);
      //
Open in  blocking read/write mode(i.e. not using O_NDELAY)

O_RDWR –open for reading and writing
O_NOCTTY –this flag ensures that the program doesn’t become the
controlling terminal for the port


      if (uart0_filestream == -1)
      {
            //ERROR - CAN'T OPEN SERIAL PORT
            printf("Error - Unable to open UART.  Ensure it is not in
use by another application\n");
      }


      struct termios options;
tcgetattr(uart0_filestream, &options);         // get the current serial
port settings

      options.c_cflag = B9600| CS8 | CLOCAL | CREAD;       // control flag
<Set baud rate, character size, CLOCAL- don’t change owner of port,
CREAD- enable receiver

      options.c_iflag = IGNPAR;                 //input flag
      options.c_oflag = 0;                      //output flag
      options.c_lflag = 0;                      //line flag
      tcflush(uart0_filestream, TCIFLUSH);
      tcsetattr(uart0_filestream, TCSANOW, &options);      //sets the
serial port setting immediately
```

```
//IGNPAR- ignore parity errors
// TCIFLUSH- flushes the input and output queue

//TCSANOW-  all changes should occur immediately without waiting for output data to finish sending or input data to finish receiving
```

sending data:

```
/* //----- TX BYTES -----
      unsigned char tx_buffer[20];
      unsigned char *p_tx_buffer;
      printf("enter char: ");
      scanf("%s",tx_buffer);

// the following while 1 can be replaced by a for loop with appropriate number of iterations

while(1){  if (uart0_filestream != -1)
      {
            int count = write(uart0_filestream, &tx_buffer[0], (p_tx_buffer - &tx_buffer[0]));
            if (count < 0)
            {
                  printf("UART TX error\n");
            }
      }} */
```

Receiving data:

```
//----- CHECK FOR ANY RX BYTES -----

      if (uart0_filestream != -1)
      {
            // Read up to 255 characters from the port if they are there
            unsigned char rx_buffer[256];
            int rx_length = read(uart0_filestream, (void*)rx_buffer, 255);
            if (rx_length < 0)
            {
                  //An error occured (will occur if there are no bytes)
            }
            else if (rx_length == 0)
            {
                  //No data waiting
            }
            else
            {
```

```
                //Bytes received
                rx_buffer[rx_length] = '\0';
                printf("%i bytes read : %s\n", rx_length,
rx_buffer);
            }
        }


}
```

we need to install g++/gcc compiler to compile the code file and make it executable

to do this , write in terminal

sudo apt-get install g++

Before executing the code it should be made sure that the serial port is not in use by another application.

To do this, write in terminal

sudo chmod a+rw /dev/ttyAMA0 // for raspbian

sudo chmod a+rw /dev/ttyUSB0 // for linux

Then making the c file executable (let the filename be uart.c and the executable file to be made be named as uart1)

To do this, write in terminal

g++ uart.c –o uart1 // (g++ can be replaced by gcc) &(uart.c should be replaced by uart.cpp if the filename is uart.cpp)

Now send code should be executed at raspberry pi terminal and receive code should be executed at linux terminal(hardware- FTDI chip) or vice versa. Care should be taken that baudrate should be the same for both.

For more reference:-

1. https://www.cmrr.umn.edu/~strupp/serial.html
2. http://www.raspberry-projects.com/pi/programming-in-c/uart-serial-port/using-the-uart