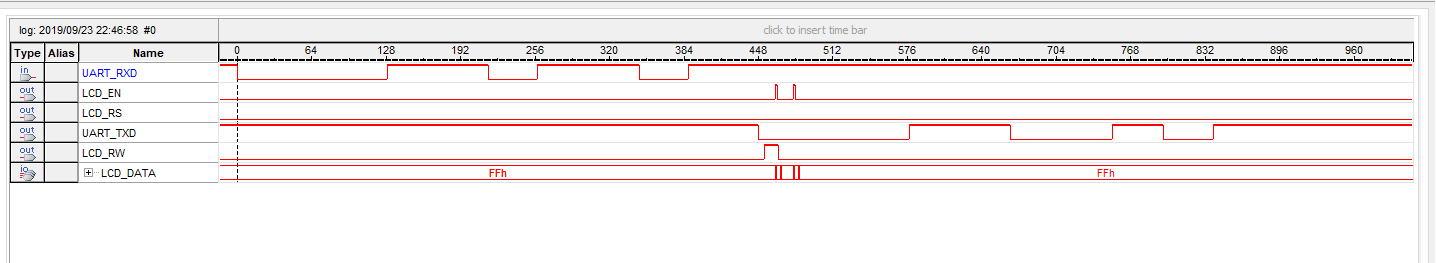
|  |  |
| --- | --- |
| ENGR 325 Lab 3  Interfacing with UART Devices.  23rd Sept 2019.  Prof Randal Brouwer | Abstract  This lab focused on working with serial communication devices (UARTs), generating a complex system using the Qsys tool, using the Signal Tap Logic Analyzer tool, and writing C language programs for the NIOS processor system using function calls provided by the Hardware Abstraction Layer (HAL) (or device drivers).  Daniel Ackuaku |

 Part 1:

#include "altera\_up\_avalon\_character\_lcd.h"

#include "altera\_up\_avalon\_rs232.h"

int main(void)

{

// declaration of global vairables for the LCD and Serial

alt\_up\_character\_lcd\_dev \* char\_lcd\_dev;

alt\_up\_rs232\_dev\* Serial;

// open the Character LCD port

char\_lcd\_dev = alt\_up\_character\_lcd\_open\_dev ("/dev/LCD");

/\* Initialize the character display \*/

alt\_up\_character\_lcd\_init (char\_lcd\_dev);

/\* Write "the DE2 board" in the second row \*/

char dummy[] = "\0"; // temporailiy houses characters that make up username

char username[] = "Daniel"; // hardcoded username

char user\_input[16];

char password[] = "qwerty1234";

char user\_pass[16];

//alt\_up\_character\_lcd\_string(char\_lcd\_dev, dummy);

/\* Open the Serial connection \*/

Serial = alt\_up\_rs232\_open\_dev ("/dev/Serial");

/\* Declaraiotion of variables needed to read and write data to LCD and PuTTy \*/

alt\_u8 rs232\_r\_data; // character array in memory to house read data

alt\_u8 \*rs232\_r\_ptr = &rs232\_r\_data; // pointer to address of read character array

alt\_u8 rs232\_w\_data1; // character array in memory to house write data

alt\_u8 \*rs232\_w\_ptr = &rs232\_w\_data1; // pointer to address if write character array

alt\_u8 rs232\_w\_data2; // character array in memory to house write data

alt\_u8 \*rs232\_w\_ptr2 = &rs232\_w\_data2; // pointer to address if write character array

alt\_u8 \*parity\_error; // parity\_error pointer for read funciton

alt\_up\_rs232\_read\_data( Serial, rs232\_r\_ptr, parity\_error);

int login\_size = 15;

int password\_size = 15;

int i = 0;

/\* The while loop runs continually the amin loop that houses the rest of the functions \*/

while(1) {

/\* Write "Username :" in the first row \*/

alt\_up\_character\_lcd\_init (char\_lcd\_dev); // clears the display

alt\_up\_character\_lcd\_string(char\_lcd\_dev, "Enter login :");

alt\_up\_character\_lcd\_set\_cursor\_pos(char\_lcd\_dev, 0, 1); //moves cursor to the 2nd row

while (rs232\_r\_data != "\r" && i < login\_size){

if ( alt\_up\_rs232\_read\_data( Serial, rs232\_r\_ptr, parity\_error) == 0) {

dummy[0] = rs232\_r\_data;

rs232\_w\_data1 = rs232\_r\_data; // data written to PuTTy

alt\_up\_rs232\_write\_data(Serial, rs232\_w\_data1); // output char to PuTTy

alt\_up\_character\_lcd\_string(char\_lcd\_dev , dummy);

user\_input[i] = dummy[0];

i++;

dummy[0] = "\0";

}

if (rs232\_r\_data == '\r') {

i = 0;

break;

}

}

//Prompts the user to continue

alt\_up\_character\_lcd\_init (char\_lcd\_dev); // clears the display

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "Login received \0");

alt\_up\_character\_lcd\_init (char\_lcd\_dev); // clears the display

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "Enter password : ");

alt\_up\_character\_lcd\_set\_cursor\_pos(char\_lcd\_dev, 0, 1);

while (rs232\_r\_data != "\r" && i < login\_size){

if ( alt\_up\_rs232\_read\_data( Serial, rs232\_r\_ptr, parity\_error) == 0) { // if a read was successfull

dummy[0] = rs232\_r\_data;

rs232\_w\_data2 = rs232\_r\_data; // data written to PuTTy

alt\_up\_rs232\_write\_data(Serial, rs232\_w\_data2); // output char to PuTTy

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "\*");

user\_pass[i] = dummy[0];

i++;

dummy[0] = "\0";

}

if (rs232\_r\_data == '\r') {

i = 0;

break;

}

}

if (username[i] == user\_input[i] && password[i] == user\_pass[i] ) {

alt\_up\_character\_lcd\_init (char\_lcd\_dev); // clears the display

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "Login Succesful: ");

alt\_up\_character\_lcd\_set\_cursor\_pos(char\_lcd\_dev, 0, 1);

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "Welcome");

return 0;

}

else {

int y = 0;

alt\_up\_character\_lcd\_init (char\_lcd\_dev); // clears the display

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "Attempt failed :(.. ");

alt\_up\_character\_lcd\_set\_cursor\_pos(char\_lcd\_dev, 0, 1);

alt\_up\_character\_lcd\_string(char\_lcd\_dev , "Try again");

for (y = 0; y < 40000 ; y++) {

alt\_up\_character\_lcd\_cursor\_off(char\_lcd\_dev);

alt\_up\_character\_lcd\_cursor\_blink\_on(char\_lcd\_dev);

}

y = 0;

}

if (i = 0) {

alt\_up\_character\_lcd\_init (char\_lcd\_dev); // clears the display

break;

}

}

}