Tutorial of Interfacing with RS232 UART

Kwan Yin Lau (group 3)
Feb 7, 2014

Introduction

This tutorial will demonstrate how to interface the RS232 UART port on the Altera DE2 board in order to send and receive characters between a simple RS232 terminal software and the DE2 board. The hardware and software components are modified from ECE 492 lab1 materials. This simple application use a switch to select the UART mode from READ or WRITE and display it on LCD. When WRITE mode on, the application on DE2 board will write a character 'A' with a new line character to the RS232 terminal in every two seconds. When READ mode on, the application will read any input from the RS232 terminal, display it on Console with one character per second.

Procedure

1. Open the lab1 project in Quartus II, select Project -> Archive Project -> Archive.
2. Copy the *.qar file generated to a new directory. Restore Archived Project to restore it in that directory.
3. Open Qsys, add a new RS232 UART interface. Select University Program -> Communications -> RS232 UART. Keep everything as default, click finish.
4. Connect the clock_reset signal to your c1 from altpll_0 (if you are not modifying from lab1, use your clk source here), connect processor data master to avalon_rs232_slave. Create Global Reset Network and Assign Base Address from System. Double click under export column on the external_interface Conduit. Click on the IRQ column to add the IRQ.
5. Generate the new design.
6. Modify your top-level *.vhd as follows:
    a. under port of entity, declare the input port
    ```
    UART_RXD   : in  std_logic;
    UART_TXD   : out  std_logic;
    ```
    b. under the component of your niosII_system port, declare
    ```
    rs232_0_external_interface_RXD : in std_logic := 'X';
    rs232_0_external_interface_TXD : out std_logic;
    ```
    c. under u0 port map, add
    ```
    rs232_0_external_interface_RXD => UART_RXD,
    rs232_0_external_interface_TXD => UART_TXD,
    ```
7. Compile the new design.
8. On any computer which has a RS232 serial port (ie: the lab machine), Download the RS232 terminal software from the link below (choose the 58KB program only version) and unzip it: http://www.compuphase.com/software_termite.htm
9. Connect the RS232 port on both DE2 board and the lab machine with a RS232 serial cable.
10. Open Termite.exe, click Settings, make sure the Port in Port Configuration is the port connected with the cable.
11. Modify the software from lab1 with reference to the following code.

```
#include <stdio.h>
#include "includes.h"
#include "altera_up_avalon_character_lcd.h"
#include "altera_avalon_pio_regs.h"
#include "altera_up_avalon_rs232.h"
#include "altera_up_avalon_rs232_regs.h"
#include "sys/alt_irq.h"
#include "alt_types.h"

/* Definition of Task Stacks */
#define   TASK_STACKSIZE       2048
OS_STK taskLCD_stk[TASK_STACKSIZE];
OS_STK taskSW_stk[TASK_STACKSIZE];
OS_STK taskRS232_stk[TASK_STACKSIZE];
OS_STK SWQ_stk[TASK_STACKSIZE];
OS_STK RS232Q_stk[TASK_STACKSIZE];

/* Definition of Task Priorities */
#define TASKLCD_PRIORITY       3
#define TASKSW_PRIORITY        2
#define TASKRS232_PRIORITY    1

#define     SW_READ            1
#define     SW_WRITE           2
#define     WRITE_FIFO_EMPTY   0x80
#define     READ_FIFO_EMPTY    0x0

OS_EVENT *SWQ;
OS_EVENT *RS232Q;
INT8U err;

/* Display Read/Write status to the LCD that is changed based on a
 signal from the Switch task */
void taskLCD(void* pdata) {
      alt_up_character_lcd_dev * char_lcd_dev;
      // open the Character LCD port
      char_lcd_dev = alt_up_character_lcd_open_dev("/dev/character_lcd_0");
      if (char_lcd_dev == NULL)
            alt_printf("Error: could not open character LCD device\n");
      else
            alt_printf("Opened character LCD device\n");
      while (1) {
            if (OSQPend(SWQ, 0, &err) == SW_WRITE) {
                  /* Initialize the character display */
                  alt_up_character_lcd_init(char_lcd_dev);

                  /* Write "WRITE" in the second row */
                  char second_row[] = "WRITE\0";
                  alt_up_character_lcd_set_cursor_pos(char_lcd_dev, 0, 1);
                  alt_up_character_lcd_string(char_lcd_dev, second_row);
```

```
            } else {
                  alt_up_character_lcd_init(char_lcd_dev);
                  /* Write "READ" in the first row */
                  alt_up_character_lcd_string(char_lcd_dev, "READ");
            }
            OSTimeDlyHMSM(0, 0, 0, 50);
      }
}
/* poll one of the switches and send a message to the
 LCD task */
void taskSW(void* pdata) {
      while (1) {
            if (IORD_ALTERA_AVALON_PIO_DATA(SWITCH_BASE) == 1) {
                  err = OSQPost(SWQ, SW_WRITE);
            } else {
                  err = OSQPost(SWQ, SW_READ);
            }
            OSTimeDlyHMSM(0, 0, 0, 50);
      }
}
/* UART task: read and write */
void taskRS232(void* pdata) {
      alt_u32 write_FIFO_space;
      alt_u16 read_FIFO_used;
      alt_u8 data_W8;
      alt_u8 data_R8;
      int enter = 0;
      unsigned p_error;

      alt_up_rs232_dev* rs232_dev;
      // open the RS232 UART port
      rs232_dev = alt_up_rs232_open_dev("/dev/rs232_0");
      if (rs232_dev == NULL)
            alt_printf("Error: could not open RS232 UART\n");
      else
            alt_printf("Opened RS232 UART device\n");

      alt_up_rs232_enable_read_interrupt(rs232_dev);

      while (1) {
            int sw = OSQPend(SWQ, 0, &err);
            if (sw == SW_WRITE) {
                  alt_up_rs232_disable_read_interrupt(rs232_dev);

                  if (enter == 0) {
                        data_W8 = 'A';
                        enter = 1;
                  } else if (enter == 1) {
                        data_W8 = '\n';
```

```
                        enter = 0;
                  }
                  write_FIFO_space = alt_up_rs232_get_available_space_in_write_FIFO(
                              rs232_dev);
                  if (write_FIFO_space >= WRITE_FIFO_EMPTY) {
                        alt_up_rs232_write_data(rs232_dev, data_W8);
                        alt_printf("write %c to RS232 UART\n", data_W8);
                  }
                  OSTimeDlyHMSM(0, 0, 1, 0);
                  alt_up_rs232_enable_read_interrupt(rs232_dev);
            }
            if (sw == SW_READ) {
                  read_FIFO_used = alt_up_rs232_get_used_space_in_read_FIFO(
                              rs232_dev);
                  if (read_FIFO_used > READ_FIFO_EMPTY) {
                        alt_printf("char stored in read_FIFO: %x\n",
read_FIFO_used);
                        alt_up_rs232_read_data(rs232_dev, &data_R8, &p_error);
                        alt_printf("read %c from RS232 UART\n", data_R8);
                  }
                  OSTimeDlyHMSM(0, 0, 1, 0);
            }
      }
}

/* The main function creates three task and starts multi-tasking */
int main(void) {
      OSInit();
      SWQ = OSQCreate(SWQ_stk, TASK_STACKSIZE);
      RS232Q = OSQCreate(RS232Q_stk, TASK_STACKSIZE);

      OSTaskCreateExt(taskLCD, NULL, (void *) &taskLCD_stk[TASK_STACKSIZE - 1],
                  TASKLCD_PRIORITY, TASKLCD_PRIORITY, taskLCD_stk, TASK_STACKSIZE,
                  NULL, 0);

      OSTaskCreateExt(taskSW, NULL, (void *) &taskSW_stk[TASK_STACKSIZE - 1],
                  TASKSW_PRIORITY, TASKSW_PRIORITY, taskSW_stk, TASK_STACKSIZE,
NULL,
                  0);

      OSTaskCreateExt(taskRS232, NULL,
                  (void *) &taskRS232_stk[TASK_STACKSIZE - 1], TASKRS232_PRIORITY,
                  TASKRS232_PRIORITY, taskRS232_stk, TASK_STACKSIZE, NULL, 0);

      OSStart();
      return 0;
}
```

Description

- LCD and switch tasks are very similar to lab 1, details will be omitted here.
- To set up and open the UART device, use the function `alt_up_rs232_open_dev`. It is similar to the steps when opening the LCD. Also enable read interrupt using function `alt_up_rs232_enable_read_interrupt`.
- In the large while loop, it pend on the queue from switch which determines READ/WRITE mode and check for the mode.
- If it is READ mode, use function `alt_up_rs232_get_used_space_in_read_FIFO` which returns the number of characters remaining in read FIFO buffer that is not yet read. If that function returns a number greater than 0, call function `alt_up_rs232_read_data` to read from the read buffer, the character read will be stored in a 8-bit variable (the pointer of that variable pass to read data function as parameter). After a character (a byte) has been read, it will be removed from the read FIFO buffer.
- If it is WRITE mode, first disable read interrupt using `alt_up_rs232_disable_read_interrupt`, re-enable it at the end of all write-related jobs. Before a write to write buffer, call `alt_up_rs232_get_available_space_in_write_FIFO` which returns number of bits available in write buffer, to make sure it greater than 8 (length of a character, also the data width of UART). Use function `alt_up_rs232_write_data` to write the prepared 8-bit character.
- Since the hardware abstract layer is provided, we do not need to modify the UART registers directly. Although enable/disable read interrupt is used, the implementation is much closer to a polling fashion. Instead of reading the read ready and write ready interrupt registers in an interrupt handler, we check the space in read/write FIFO buffers every time (every second here) we want to do a read/write operation through UART. To minimize the disadvantage of polling, we can modify the implementation in the future, pending on queues or semaphores, such that it only "poll" the UART when needed.
- Noted that there are two UART components available in Qsys, we use RS232 UART in University Program instead of UART (RS-232 Serial Port) under Serial of Interface Protocols. We don't really sure what the difference between them, but we never get the other one works...

Source Documentation and Reference

RS232 UART: RS232 UART for Altera DE-Series Boards – Altera University Program
file:///C:/altera/12.1sp1/ip/University_Program/Communications/altera_up_avalon_rs232/doc/RS232.pdf
altera_up_avalon_rs232_regs.h
altera_up_avalon_rs232.h
PIO: Chapter 10 of the Embedded Peripherals IP User Guide – Altera Quartus® II design software
ug_embedded_ip.pdf
altera_avalon_pio_regs.h
LCD: 16x2 Character Display for Altera DE2-Series Boards – Altera University Program
ug_embedded_ip.pdf
altera_up_avalon_character_lcd_regs.h
altera_up_avalon_character_lcd.h
RS232 terminal software: CompuPhase Termite http://www.compuphase.com/software_termite.htm
Hardware modified from ECE 492 lab 1.
Code modified from Hello MicroC/OS-II template from Nios II Software Build Tools for Eclipse and ECE 492 lab 1.