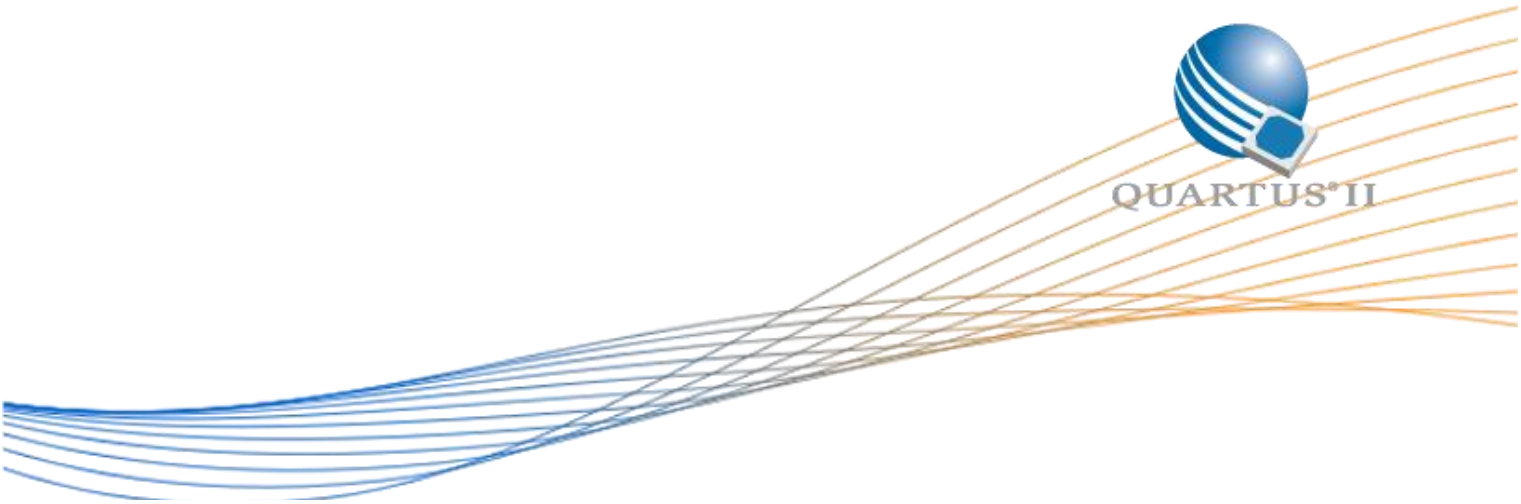


UART RS-232 Maximum Baud Rate Design Example Tutorial

Date: 1 December 2016

Revision:1.0

Table of Contents

1. Introduction

This example is a test functionality for UART RS-232 Serial Port IP which contains a NIOS® II processor and Dual UART RS-232 IP. The design example implements a basic UART RS-232 functionality of Variable Baud Rate On real-time basis. This means the developer can set the required Baud Rate of data transfer from NIOS II Application. Furthermore, this design demonstrates a standard method of developing a UART Application with NIOS II.

2. Prerequisite

You are required to have knowledge in instantiating and developing a system with a Nios II processor. Altera recommends that you go through the online tutorials and training materials provided on Altera's website before using this design example.

Related information:

- Nios II Gen2 Software Developer's Handbook
- Embedded Peripherals IP User Guide

3. Requirements

The hardware and software requirements for the design example are:

- Cyclone V DE0-Nano SoC board.
- Altera Quartus Prime version 16.1 with Nios II EDS.
- USB cable for programming the device.
- Two Jumper wire (female).

4. Design Example Files

This design example consists of hardware image and Nios II software. The Application hardware and software images include simple design to run the Dual UART application.

File Name	Description
Output_files/uart.sof	Main Hardware image of the design
Software/software.zip	The software files of NIOS II application used in this example
Top.v	This is the top-level file of the design
Uart.qsys	Qsys file of the NIOS II system

5. Block Diagram

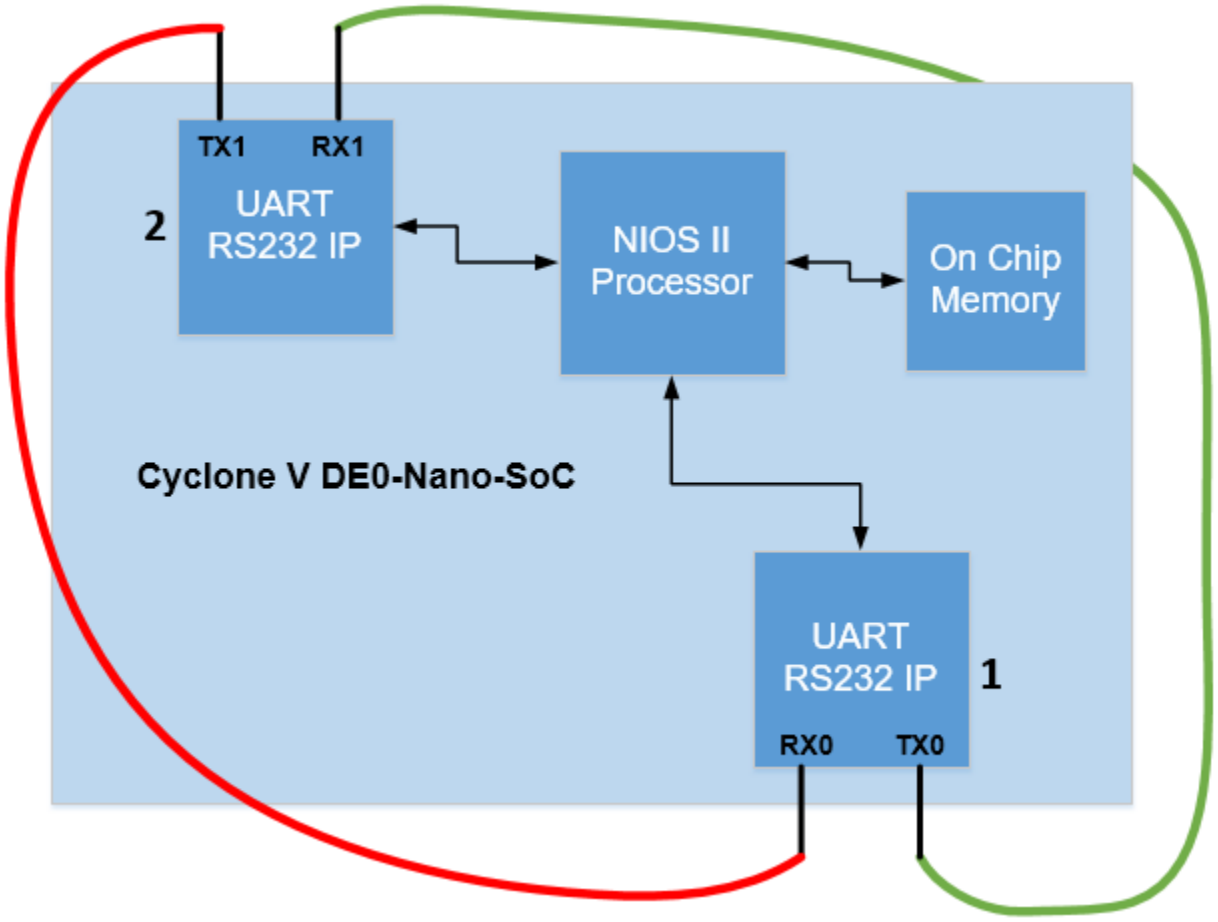
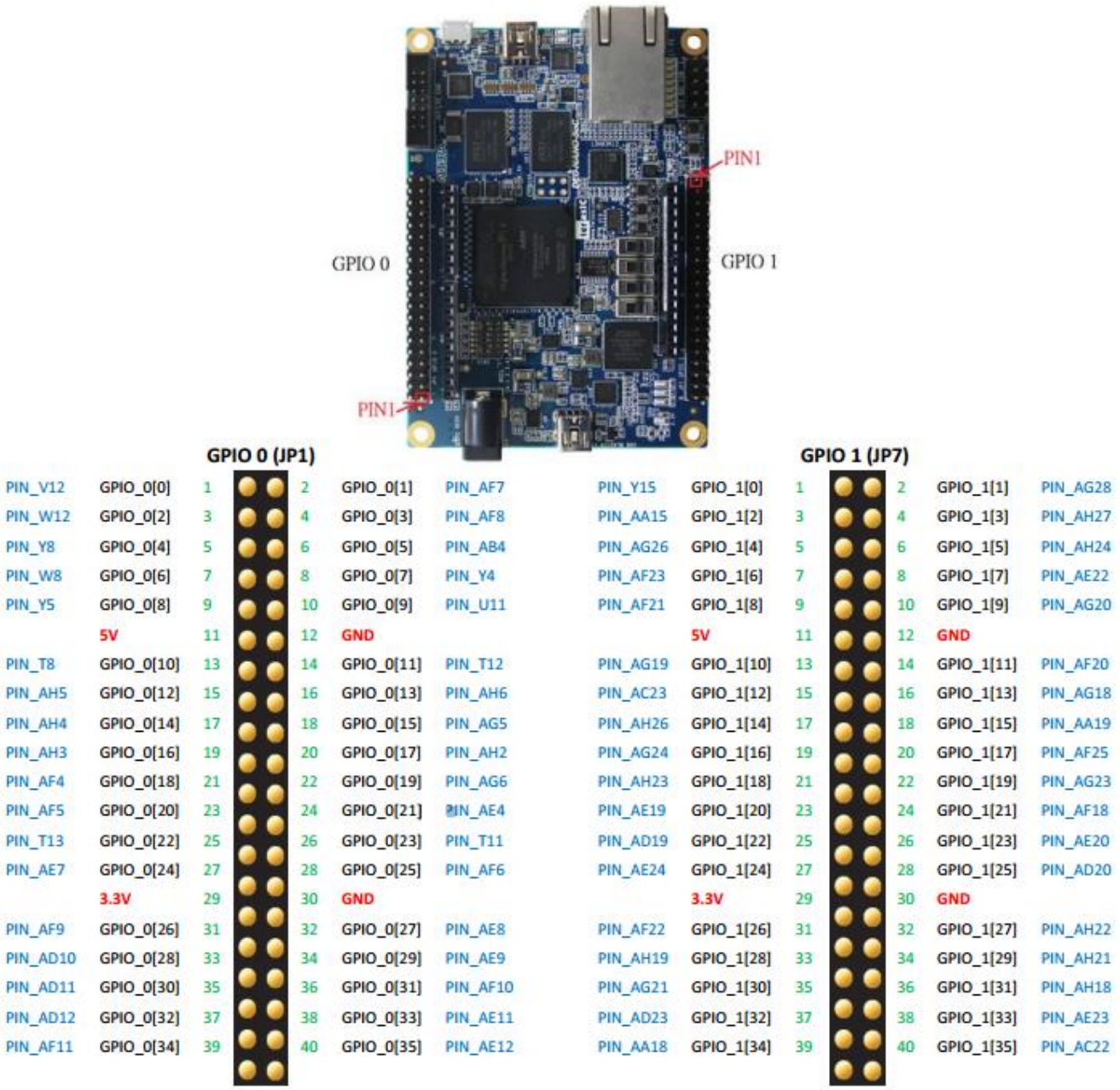


Figure 1: UART RS232 Design Diagram

Figure 1 shows the various IP blocks within the design example. Details of the connection between each IP blocks can be found in the included Qsys file for this design example. From Figure 1, three main steps can be summarized:

1- The UART 1 will send some random characters through TX0.
2- The UART 2 will receive the sent characters from UART 1 through RX1.
3- NIOS II processor will print out some data on console.

4- The colored lines in Figure 1 represent the jumper wires (female) that connect GPIO 0 Pin 1 with GPIO 1 Pin 1. Kindly refer to Cyclone V DE0-Nano-SOC user guide (Page 26). Figure 3-18 from this document can be seen in Figure 2.



PIN_V12	GPIO_0[0]	1	2	GPIO_0[1]	PIN_AF7		PIN_Y15	GPIO_1[0]	1	2	GPIO_1[1]	PIN_AG28	
PIN_W12	GPIO_0[2]	3	4	GPIO_0[3]	PIN_AF8		PIN_AA15	GPIO_1[2]	3	4	GPIO_1[3]	PIN_AH27	
PIN_Y8	GPIO_0[4]	5	6	GPIO_0[5]	PIN_AB4		PIN_AG26	GPIO_1[4]	5	6	GPIO_1[5]	PIN_AH24	
PIN_W8	GPIO_0[6]	7	8	GPIO_0[7]	PIN_Y4		PIN_AF23	GPIO_1[6]	7	8	GPIO_1[7]	PIN_AE22	
PIN_Y5	GPIO_0[8]	9	10	GPIO_0[9]	PIN_U11		PIN_AF21	GPIO_1[8]	9	10	GPIO_1[9]	PIN_AG20	
	5V	11	12	GND				5V	11	12	GND		
PIN_T8	GPIO_0[10]	13	14	GPIO_0[11]	PIN_T12		PIN_AG19	GPIO_1[10]	13	14	GPIO_1[11]	PIN_AF20	
PIN_AH5	GPIO_0[12]	15	16	GPIO_0[13]	PIN_AH6		PIN_AC23	GPIO_1[12]	15	16	GPIO_1[13]	PIN_AG18	
PIN_AH4	GPIO_0[14]	17	18	GPIO_0[15]	PIN_AG5		PIN_AH26	GPIO_1[14]	17	18	GPIO_1[15]	PIN_AA19	
PIN_AH3	GPIO_0[16]	19	20	GPIO_0[17]	PIN_AH2		PIN_AG24	GPIO_1[16]	19	20	GPIO_1[17]	PIN_AF25	
PIN_AF4	GPIO_0[18]	21	22	GPIO_0[19]	PIN_AG6		PIN_AH23	GPIO_1[18]	21	22	GPIO_1[19]	PIN_AG23	
PIN_AF5	GPIO_0[20]	23	24	GPIO_0[21]	PIN_AE4		PIN_AE19	GPIO_1[20]	23	24	GPIO_1[21]	PIN_AF18	
PIN_T13	GPIO_0[22]	25	26	GPIO_0[23]	PIN_T11		PIN_AD19	GPIO_1[22]	25	26	GPIO_1[23]	PIN_AE20	
PIN_AE7	GPIO_0[24]	27	28	GPIO_0[25]	PIN_AF6		PIN_AE24	GPIO_1[24]	27	28	GPIO_1[25]	PIN_AD20	
	3.3V	29	30	GND				3.3V	29	30	GND		
PIN_AF9	GPIO_0[26]	31	32	GPIO_0[27]	PIN_AE8		PIN_AF22	GPIO_1[26]	31	32	GPIO_1[27]	PIN_AH22	
PIN_AD10	GPIO_0[28]	33	34	GPIO_0[29]	PIN_AE9		PIN_AH19	GPIO_1[28]	33	34	GPIO_1[29]	PIN_AH21	
PIN_AD11	GPIO_0[30]	35	36	GPIO_0[31]	PIN_AF10		PIN_AG21	GPIO_1[30]	35	36	GPIO_1[31]	PIN_AH18	
PIN_AD12	GPIO_0[32]	37	38	GPIO_0[33]	PIN_AE11		PIN_AD23	GPIO_1[32]	37	38	GPIO_1[33]	PIN_AE23	
PIN_AF11	GPIO_0[34]	39	40	GPIO_0[35]	PIN_AE12		PIN_AA18	GPIO_1[34]	39	40	GPIO_1[35]	PIN_AC22	

Figure 2: CV DE0-NANO-SOC GPIO Pin Arrangement (originally from here)

6. Experimental Setup

6.1 Generate Programming Files

First, follow the steps on Design Store webpage to prepare the design template in Quartus Prime software. The master files are provided together with the design example. You may proceed to program the FPGA run the design without any recompilation or programming file generation. The default programming file “output_files\uart.sof” can be used to configure the FPGA with the hardware image of this design. If you intend to modify the design, you need to generate qsys and compile the design based on your changes.

6.2 Program Hardware Image Into FPGA

To program the hardware image and Nios II firmware into the board, perform the following steps:

1. Connect the USB Blaster cable from PC to the on-board USB Blaster II connector and power up the board.
2. Open the “Programmer” in Quartus Prime software, then click “Hardware Setup…” and select “USB-BlasterII”.
3. Click “Auto Detect” and select “5CSEMA4”.
4. Right-click on the selected device “5CSEMA4” and select “Change File”.
5. Open the programming file “output_files \uart.sof”.
6. Check the “Program/Configure” checkbox for “5CSEMA4”.
7. Click “Start” to start programming.
8. Make sure the download is successful.
9. Power cycle the board.

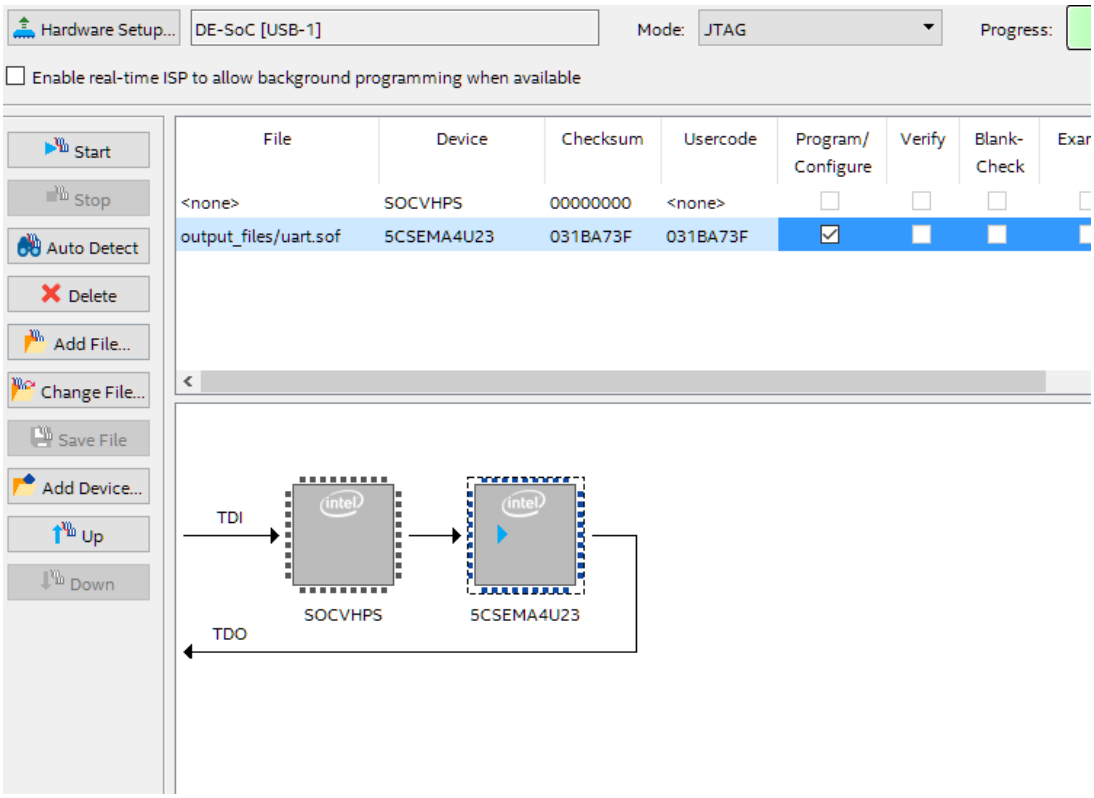


Figure 3: Programmer window

6.3 Run The Design

To setup the environment for Nios II software modification and recompilation, perform the following steps:

1. Extract archive files “software\software.zip” to folder “software\UART_RXTX” and “software\ UART_RXTX_bsp”.
2. Open the “Nios II Software Build Tools for Eclipse” from “Tools” menu in Quartus Prime software.
3. Go to “File > Import…”, then select “Import Nios II Software Build Tools Project” in “Nios II Software Build Tools Project” folder and click Next.
4. Browse to “software\ UART_RXTX” folder in “Project location” field.
5. Enter “UART_RXTX” in “Project name” field, then click “Finish”.
6. Repeat step 3 to step 5, but change the “Project location” to “software\ UART_RXTX_bsp” folder and change the “Project name” to “UART_RXTX_bsp”.
7. Repeat steps 2 to 6 for Application_1 and Application_2 folders.
8. You should see the “UART_RXTX” and “UART_RXTX_bsp” directory in “Project Explorer” as shown in Figure 4. The environment is now ready for modification.

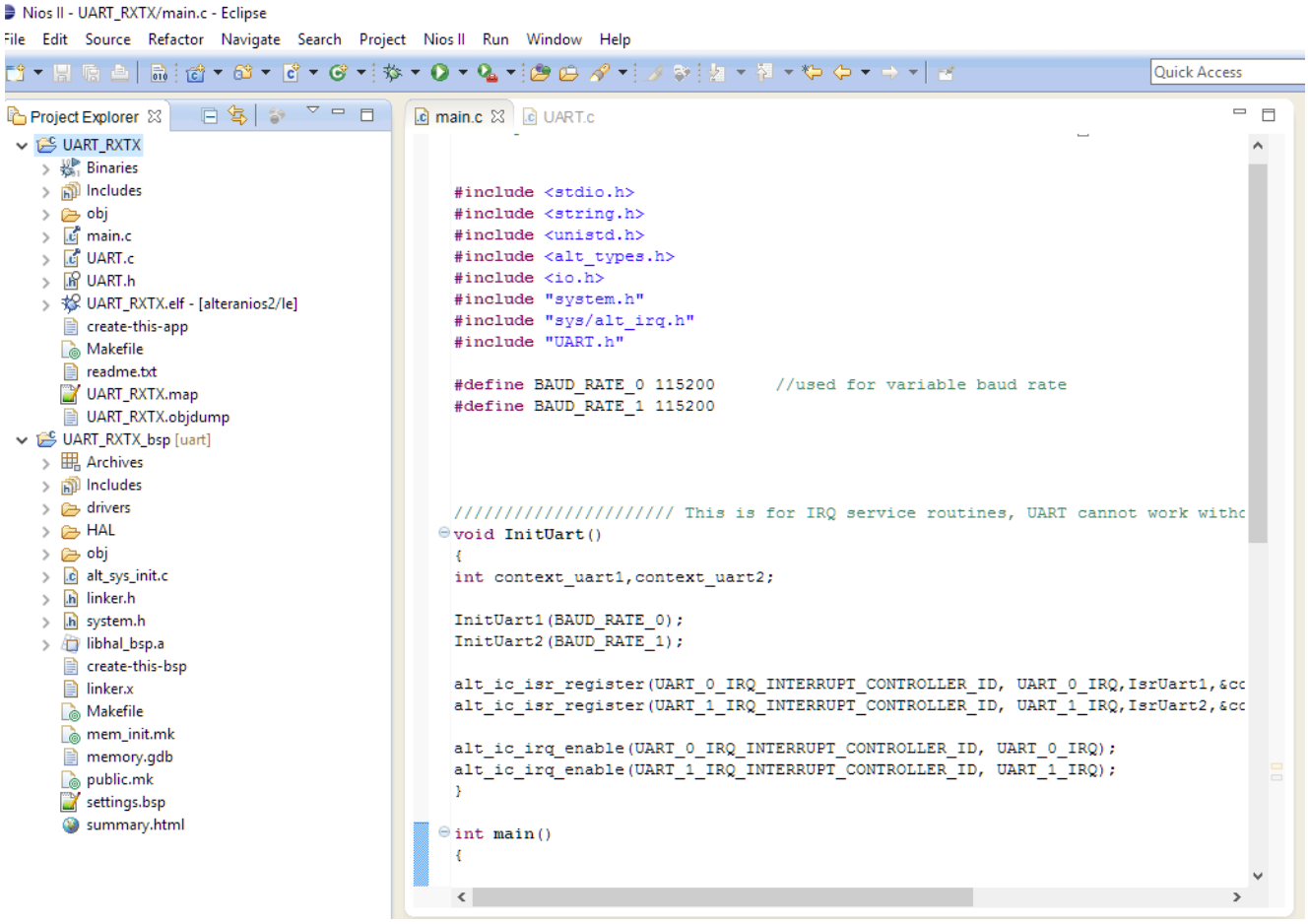


Figure 4: Project Explorer window

The software application of this design contains two main files:

1- **Main.c**
   This is the main C file of the application. It contains the random data generation, send, and print out.
2- **UART.c**
   This is the main UART functions for read/write operations. This file contains RX and TX buffers, head and tail for data transfer, driver initialization, Interrupt Service Routine (ISR) for UART, data transmission and collection, and registers check procedures.

After you build the UART_RXTX application, you can download the .elf file. Please make sure your FPGA is already configured with the uart.sof file, check Figure 2.
Once the application runs, you will notice few messages come out from NIOS II eclipse console. These messages to show the correct receiption of characters from UART 0 TX0 to UART 1 RX1, as seen in Figure 5.

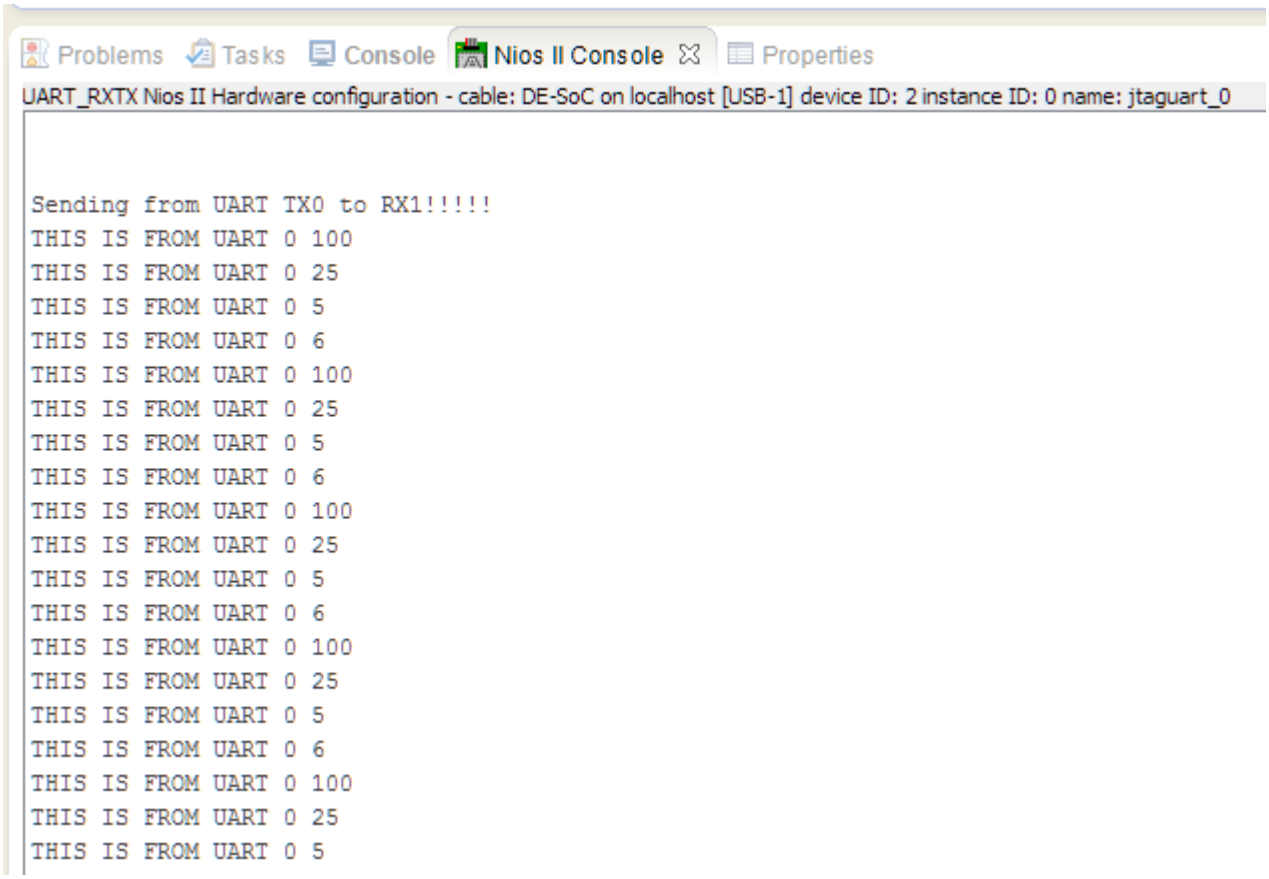


Figure 5: NIOS II eclipse console view

7. Revision History

Date	Revision	Description
10 December 2016	1.0	Initial Release