# Lab 5 - Serial Output Port

## Introduction

In this lab you will design a serial output interface capable of transmitting 8-bit characters at 9600 bps. The interface has an 8-bit data output port and a 1-bit status input port that are accessible over the PC-104 bus.

The design consists of 7 parts:

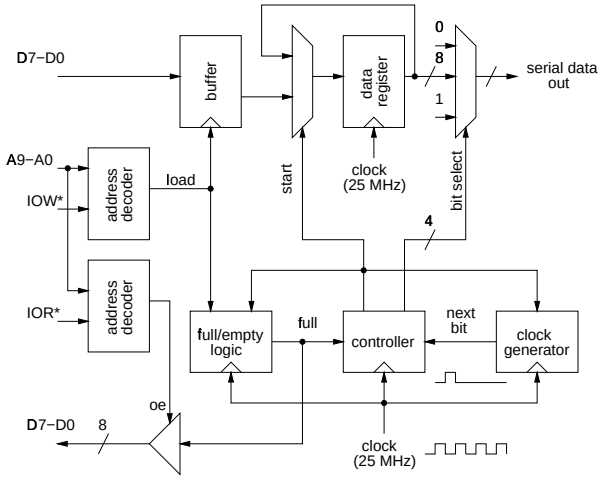
- an 8-bit output data buffer with address decoder and control logic
- a 1-bit input (status) port with address decoder and control logic
- a 1-bit buffer-full flip-flop and control logic
- an 8-bit transmit data register and control logic
- a controller (an 11-state state-machine)
- a transmit data-bit multiplexer
- a 9600 Hz clock generator

You may be able to re-use the output port, the input port and possibly the clock generator from previous labs.

You will also write an 8086 assembly-language program to output your name and student number over the serial port. You will run the program on the lab SBC and display the output of the serial port using a (second) terminal emulator program running on the lab PC.

## Hardware Description

The diagram below shows the internal structure of the serial interface:




### Data Buffer

This 8-bit register is loaded when the CPU writes to I/O port 220H. A rising edge on IOW* when the address bus contains 220H causes the contents of the data bus to be loaded into the buffer. This is the *only* register that is not clocked by the 25.175 MHz clock. The address decoder logic generates the active-low `load` strobe which is used to monitor the buffer status.

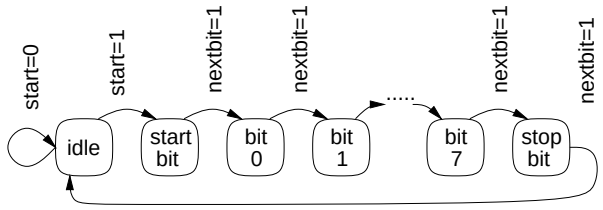
### Buffer Full/Empty Status

This circuit's `full` output indicates that the buffer register has been written by the CPU but that the buffer register has not yet been transferred to the transmit data register. `full` is set on the first clock edge after the rising edge of the `load` signal and is reset when `start` is active.

In order to to detect the rising edge of `load` you will have to use the same logic as in the previous lab (i.e. compare the current value of `load` to its previous value). There will be several clock cycles while `load` is asserted.

### Controller

The diagram below shows the state transition diagram for the controller and the value placed on the serial data output during each state (there should be no state transitions for conditions other than those shown):




While a character is being transmitted the controller causes a space (high, VHDL '1'), one of the eight data bits or a mark (low, VHDL '0') to be placed on the serial data output.

The `start` signal is asserted when the buffer is full and the transmission of the previous character is complete (i.e. when `full` is asserted and the state machine is in the idle state).

Pick a state encoding that represents the idle state as all-zero state register contents to avoid the need for additional reset logic.

### Bit Clock Generator

The bit clock generator creates a 9600 Hz signal that advances the controller state machine. To generate a 9600 Hz signal this circuit counts from 0 to $25,175,000/9,600 - 1 = 2,621$. Note that the `nextbit` output should be asserted on the *last* clock period (i.e. when the count is 2,621). The clock generator's counter is reset by the `start` signal to synchronize the bit clock to the start of the character transmission.

### Status Input Port

The CPU can read this 8-bit register at I/O port 221H. During a read from this address (a low level on IOR* when the address bus contains 221H) the `full` bit is placed on the most significant bit of the data bus with the other bits set to zero. Otherwise the bus driver is left in high-impendance mode.

### Transmit Data Register

This 8-bit register is loaded from the buffer if the `start` signal is asserted.

### Serial Data Multiplexer

This 10-to-1 multiplexer sets the serial data output to be a start bit (space, H, VHDL '1'), one of the eight data bits in the transmit data register, or a stop bit or idle level (mark, L, VHDL '0'), depending on the value of `bitselect`. Note that the RS-232 standard requires that a logic 0 data bit should be output as a high logic level (space, H, VHDL '1').

## Pre-Lab Assignment

Before the lab you must write, assemble and test (to the extent possible) an 8086 assembly-language program that does the following:

- initializes a pointer to the first character of an ASCII string consisting of your name and student number
- returns control to DOS if the current character is zero
- waits until the status bit indicates the buffer is empty
- writes the current character to the output port
- increments the pointer to the next character
- loops back to step 2

You must also design the circuit and test it by simulating its operation. Modify your code to assume the clock frequency is 19200 Hz rather than 25.175 MHz when simulating the operation of your circuit. Your simulation should demonstrate a status read that returns "not-full" status, a data write, a second data read returning "not-full", a second data write and a third data read showing "full" status. Allow the simulation to continue for at least 22 clock cycles after the first write to view the start bit, the data bits and the stop bit being transmitted for the first value written. Verify that the polarity and bit order are correct.

The TA will ask to see your assembly listings, the VHDL code and the simulation waveforms at the start of the lab. You must hand in the simulation results and the assembly-language program even if you are unable to complete the lab.

## Lab Procedure

Connect the PC-104 address and data bus signals, IOR* and IOW* to the FPGA pins on the interconnect board as described in the previous labs. Double-check your connections and turn on the power.

Connect the serial data output to the PC's second serial interface using the cable supplied.

Compile your VHDL code if you haven't already done so, and configure the FPGA as described in the previous lab.

Assemble and link your assembly code if you haven't already done so. All of your files should be stored in the `c:\max2work` directory.

Run two copies of the Windows Hyperterm program. The first should connect to the SBC as usual and be used to download and run your program. The second terminal emulator program should be set for a direct connection to COM2 so that you can monitor the output of your program. When you run your program you should see your name and student number displayed on the second terminal emulator window.

When your device is working properly ask the TA to check your work. He will make sure your device works as required and ask you one or two questions to verify your understanding of the material.

## Report

Submit a short report with a written description of your circuit. Include a block diagram showing the connections between the PC-104 bus, the FPGA the LED and the pushbutton, a listing of your assembly-language program, the VHDL code and a printout of the simulation waveforms that demonstrate correct operation of your device.

## Optional Features

To get bonus marks you can improve the design by adding add a control register (at address 221H) that selects the following options:

- different bit rates (e.g. 57600, 38400, ... 300 bps)
- different word lengths (e.g. 5, 6 ... 8 data bits)
- an optional parity bit
- an optional second stop bit