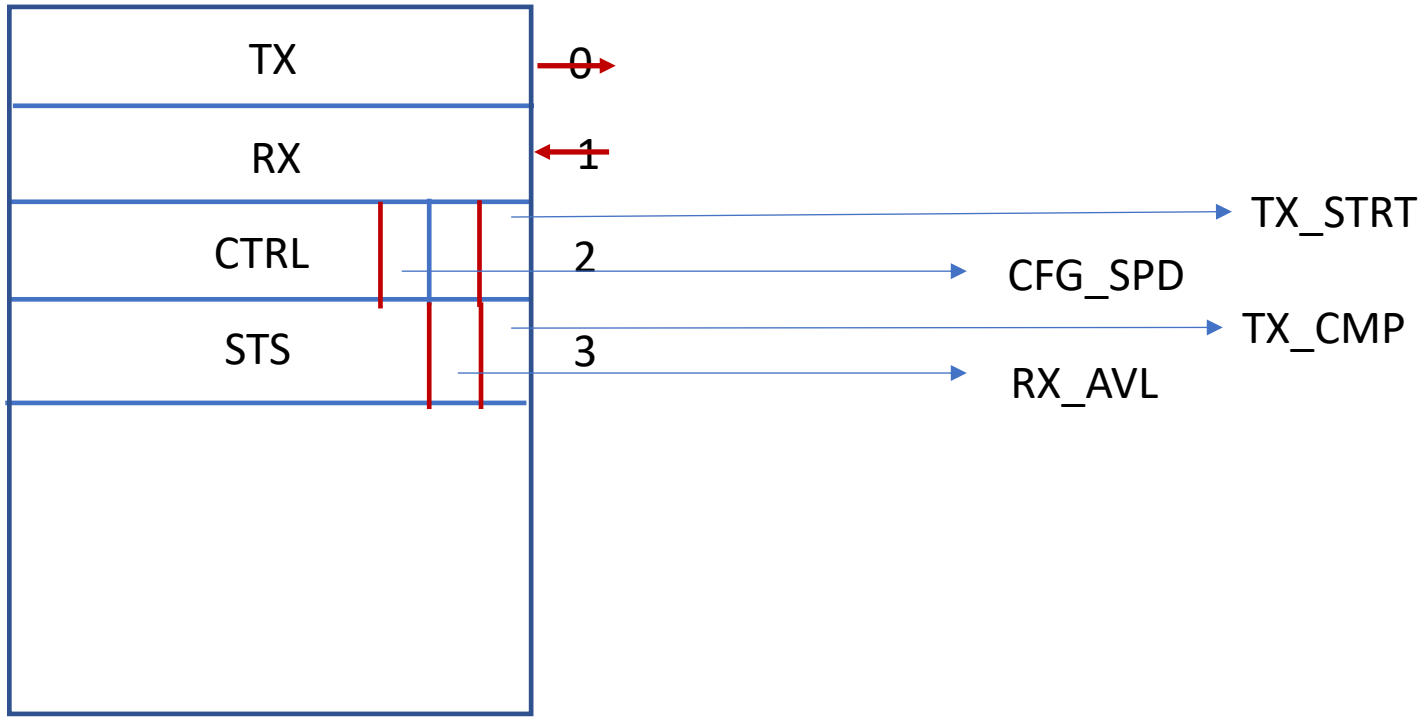
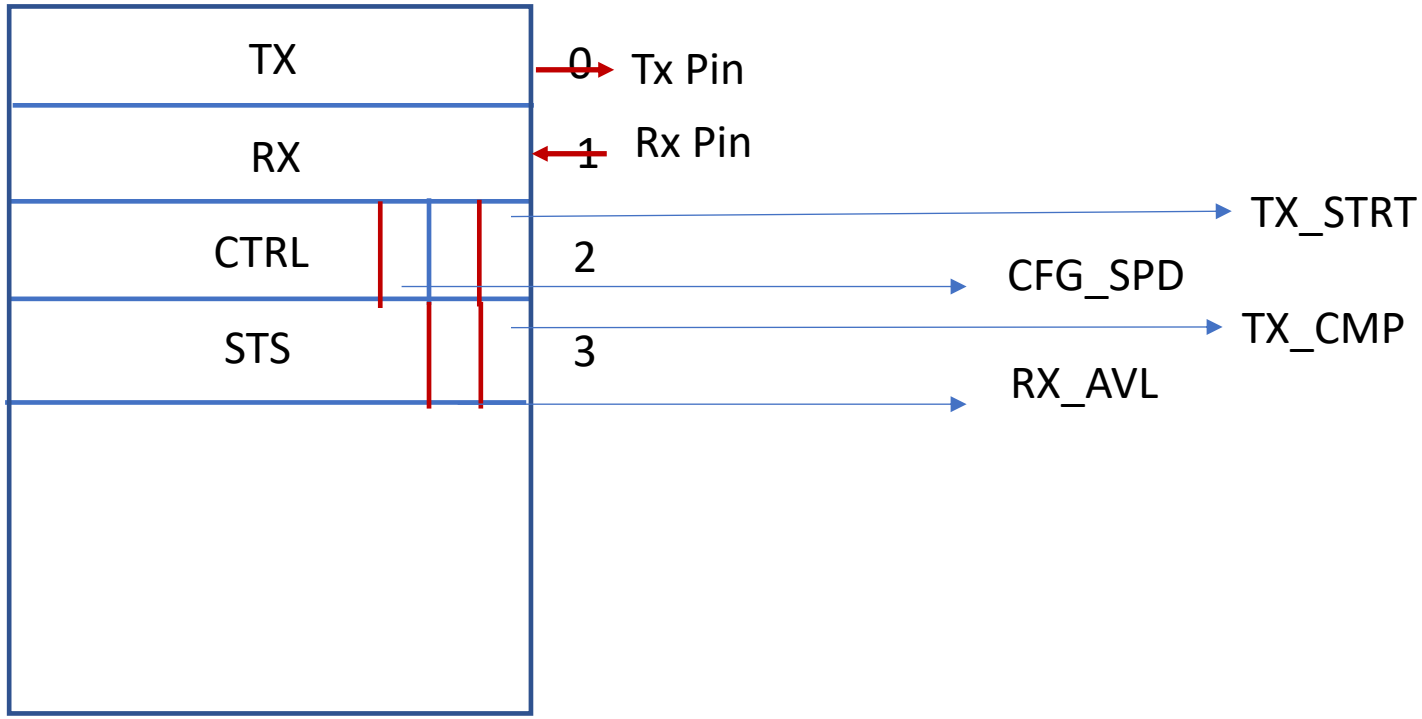
Simple Serial Device and Driver

Simple Serial Device - Description

- The device is a simplified variant of UART, that can send and receive data serially. All the registers in the device should be at least have one byte size
- The TX pin of the device is connected to a register called TX that is at offset 0
- The RX pin is connected to a register called RX that is at offset 1.
- There is register called CTRL at offset 2. This register has two “fields”. One is a one bit field called TX_STRT and the other is a two bit field called CFG_SPD.
- Finally there is a register called STS at offset 3. This is register has two fields TX_CMP and RX_AVL.



A Sample Implementation



The protocol

- The transmit protocol is:
  - Two devices will be connected using a cable that provides at least two wires to handle the Tx and Rx between the two devices
  - Both the devices should be configured to operate at the same speed (currently the supported baud rates are: 2400, 4800, 9600, 19200). Other speeds may be supported in future.
  - Transmission of data will happen after at least one byte of data is written into the TX Register and the TX_STRT bit in the CTRL register is set to 1 by the software.
  - As soon as the data is transmitted, the TX_CMP bit in the STS register will be set to 1 by the hardware.

The protocol

- The receive protocol is:
  - As soon as data is received by a device, it will be stored in the RX Register by the hardware.
  - The hardware will also set the RX_AVL bit in the STS register as soon as at least one byte of data is available in the RX register.

- Configuration:
  - Since the protocol supports 4 different speeds, at least two bit will be provided in the CTRL register, for configuring the speed of data transfer.

Simple Serial Device - Setup

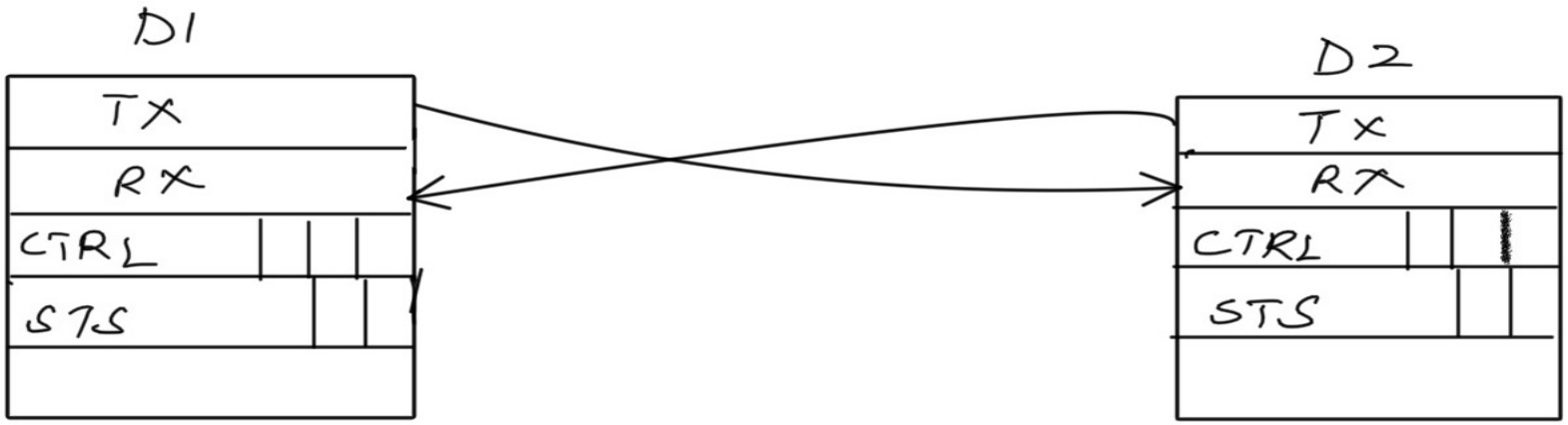
- There is only one thing that can be configured in this simple device – the speed at which data will be transferred.
- This configuration is done by writing different values into the two bit CFG_SPD field in the CTRL register. The baud rate will be set as follows:

Field value	Baud rate
0	2400
1	4800
2	9600
3	19200

Simple Serial Device – TX and RX

- To send data, the one byte data is written to the TX register and then TX_STRT bit has to be set to 1.
- When the device completes transmitting the one byte that was in the TX register it will set the bit TX_CMP in the STS register
- Whenever the device receives a byte of data that will be stored in the RX register and the RX_AVL field of the STS register will be set

Setup diagram



The two programs – P1 and P2

- The system comprising of two computer systems having the one instance each of the simple serial device is meant to be an ”echo” system – P1 running on S1 will send some data over the simple serial bus and P2 running on S2 is supposed to receive it and echo it back to P1.
- What all should be done as part of P1 and P2?

P1

1. Configure the transfer speed to the desired value by writing 0/1/2/3 to the CFG_SPD field of the CTRL register (located at offset 2)
2. While there is data to be transferred {
    1. Write one byte data to the TX register (located at offset 0)
    2. Write 1 to the STRT_TX field of the CTRL register
    3. Loop and read the TX_CMP field of the STS register (located at offset 3) until it becomes 1
    4. Loop and read the RX_AVL field of the STS register until it becomes 1
    5. Read one byte data from the RX register (located at offset 1)

}

P2

1. Configure the transfer speed to the desired value by writing 0/1/2/3 to the CFG_SPD field of the CTRL register (located at offset 2)
2. While there is more data {
    1. Loop and read the RX_AVL field of the STS register until it becomes 1
    2. Read one byte data from the RX register (located at offset 1)
    3. Write one byte data to the TX register (located at offset 0)
    4. Write 1 to the STRT_TX field of the CTRL register
    5. Loop and read the TX_CMP field of the STS register (located at offset 3) until it becomes 1

}

Questions

- How do we write to or read from registers?
- Is it a good idea to mix up the application and device access as is done in this example?
- If not what is a better approach?
- And can the better version be improved upon?