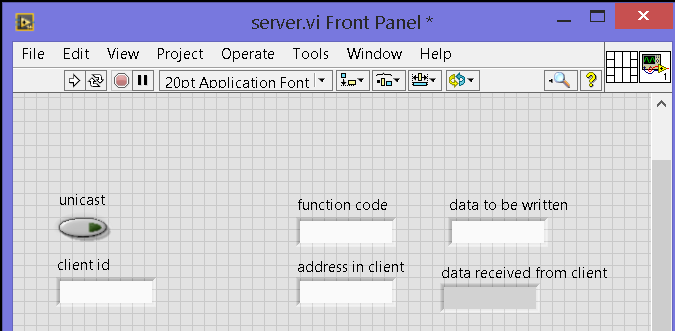
In this VI I have tried to implement the MOD bus server which can unicast messages to a particular client among multiple clients.

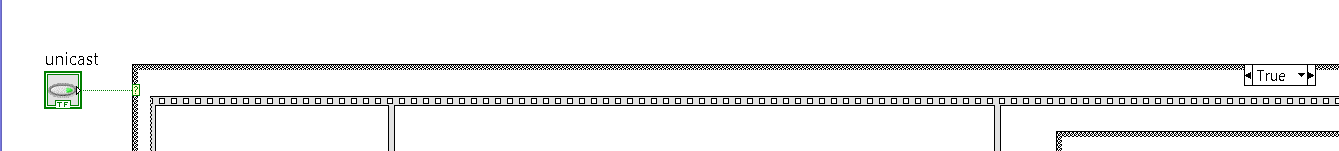
(I have tried to implement the data frame and operation of MOD bus.)



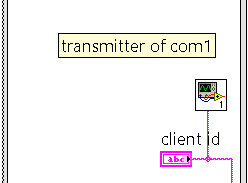
The control panel has the following controls and indicators

* Unicast – further the program can be modified to broadcast. So I have provided a selector switch
* Client id – the 7 bit name of the client. (Client Id is zero if broadcasting mode)
* Function code- to mention weather the server wants to read bit or register or number of registers (depending upon the need) or write to the client memory.
* Address in client-Address in the client in which data is to be written or to be read from.
* Data to be written
* Data read from client.

Let us go to the block diagram (I suggest you to look in the actual block diagram in the VI file along with the upcoming diagrams)

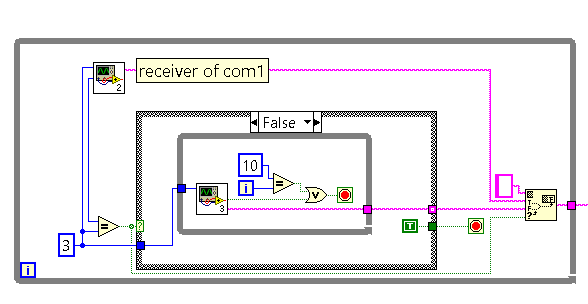


If the switch unicast is turned ON then the communication process started.



Then the server sends the client ID through serial port.

The subVI shown in the diagram is the serial transmitter VI file.



After the transmission of client id it is important to verify whether the client is available for communication or not.

For that purpose we are receiving the serial data in the same com port.

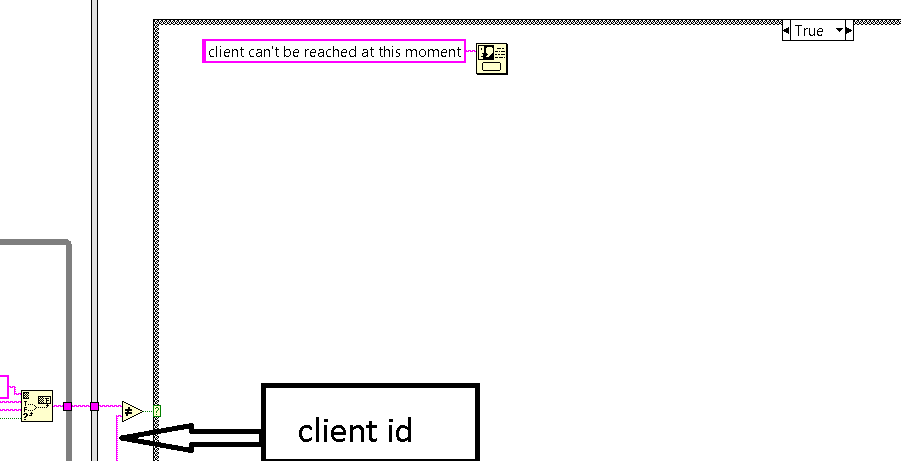
The length of the client id is 3(conventionally 1-278)

If the client id received in the first attempt, then the null constant is appended with the client Id and while loop is ended.

If the first attempt failed then the program attempts multiple receiving process (the subVI used here is SUB.VI file) till 3 bit character id is received.

The receiving attempts is limited by a constant (here 10) to avoid infinite looping.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Consider the client we want to communicate is busy in some other task or an unexpected client responds to the request.

(The received client id is different from the client Id which we expect) then the user should be intimated with an error. For that purpose I have used a one button dialog box.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

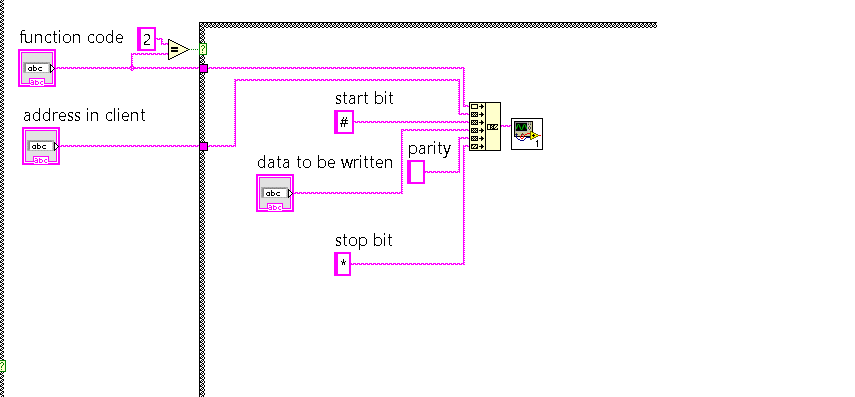
If both the client ids are same then the CORE COMMUNICATION process starts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

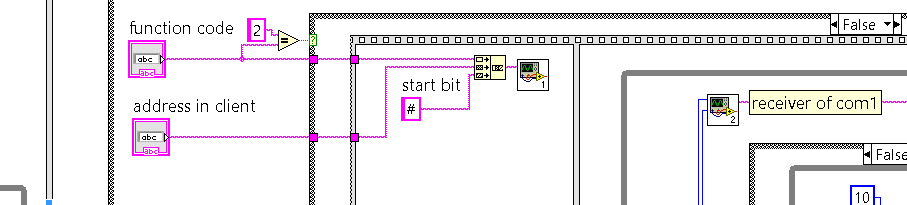
Function code- here we are transmitting or receiving only one bit (1 for transmission, 2 for reception)

In the below diagram we are sending data.

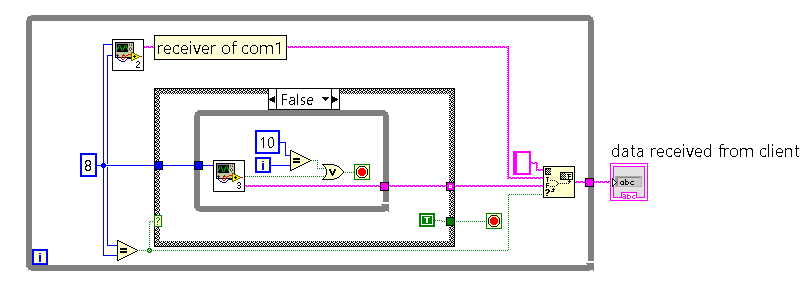
Here we use # as start bit of message and \* as stop bit of message.



If we choose to receive data from client memory.



The function code followed by the memory address in client is transmitted.



Finally the 8 bit data (for easy understanding we are neglecting the bytes, words, etc.) is received from the client using the same methodology used to receive client id.

Then the data received is directly (for simplicity we are not implementing CRC error correction) displayed to the user.