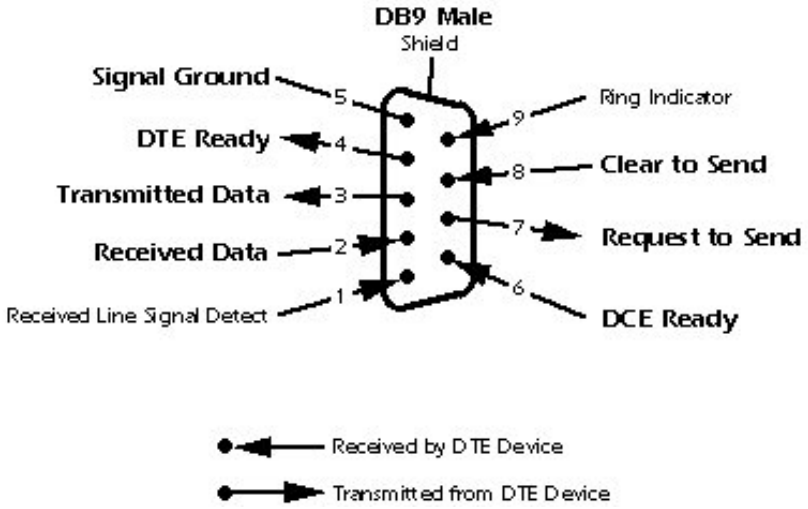
ELE306 Spring2008 LAB6

Serial Communication (RS-232/EIA-232)

RS-232, or EIA-232 from Electronic Industries Association, has been the standard for serial communication for decades (known simply as the serial port). Although it has been replaced by USB and i-Link, etc., its basic principle is still the foundation of any serial communication. There are two types of connectors, DB25 and DB9, for the 25-pin and 9-pin connectors, respectively. This lab will use the DB9 connector, as shown here. Since the PC is also using DB9, this enables us to connect directly to PC as well as another UP-2 board.



There are two types of devices that may connect to RS-232: DTE (Data Terminal Equipment), such as PC, and DCE (Data Circuit-terminating Equipment, or Data Communication Equipment), such as modem. In our case, the serial port to be implemented on our FPGA board will assume the role of DTE. Since PC is also a DTE, we can use the same cable to connect from FPGA board to PC or from FPGA board to another FPGA board. To connect between two DTE devices, we use a "null-modem" cable, which means cross-over so input pins from one side are connected to the receiving pins on the other. Specifically, pin2 of one side is connected to pin 3 on the other side. The only exception is of course pin5, the ground. A regular serial cable, or "modem cable" will have straight through connections.

Exercise #1

- Obtain "serial.qar" from the class ftp site.
- Program the FPGA from the serial project.
- The example circuit uses "9600 8N1" where 9600 refers to the baud rate (the raw transmission rate or bit per second) and 8N1 denotes that the data is 8-bit long with "N"o parity and just one stop bit. This can be seen from the "transmitter.vhd" and "receiver.vhd". The RS-232 can go even higher speed; however, since we are sending 0V-5V signals, it has to be this slow for the 6-ft cable.
    - RS-232: -3V to -12V for logic 1 and +3V to +12V for logic 0
    - This exercise: 0V for logic 1 and +4 to +5V for logic 0

- NOTE: The PS-2 port, as we have seen in the keyboard interface circuit, is similar to RS-232. Each data item is sent by: one start-bit, 8-bit data with least significant bit first, the parity bit (no parity bit in our case) and the stop bit.
- **Transmitter.vhd**: This is the transmitter that sends the data out through RS-232 port. First, the 25MHz clock is divided by 2618 to obtain a (approximately) 9.6KHz clock. The transmission is quite straightforward, as described above.
- **Receiver.vhd**: Receiving RS-232 data is tricky. As you've seen in the transmitter example, you don't need to be running exactly at 9.6KHz. Also, the data you are receiving through RS-232 is not synchronizing with your local clock. To ensure reliable reception, the receiver is running at 16X clock (and thus the 25MHz clock is divided by 164 here). Remember that, each one "bit" of the incoming data will take 16 clock cycles of this receiver.
- **Exercise**: Use the DB-9 null-modem cable to connect to the Linux PC. Open a terminal window on Linux desktop and run "minicom". The minicom has been setup so the default should work with this exercise. In case the communication is not established, you should check the minicom parameters to make sure: Use "ctrl-A" follow by "Z" to enter the minicom menu. You should choose 9600 baud rate, 8N1, and no hardware flow control. The key strokes on your FPGA board are first converted to ASCII codes and then send to "minicom" via serial port. When typing on your Linux PC with "minicom" running, the ASCII codes are sent to your FPGA also via serial port and show up on the seven-segment displays. Below is the full ASCII (uses only seven bits) chart. Find the portion where the key codes are converted to ASCII codes. Note that, not all the codes are converted in this exercise. The non-converted codes are shown as "#".

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Exercise #2

- Obtain "pong.qar" from the class ftp site.
- Program the UP-2 board and see a pong game demonstration.
- **Exercise**: Figure out from the VHDL code and write answers in your report:
  - How the ball and paddle speed is controlled?
  - How to write to RAM reliably?
  - How to paint and erase objects using Video_RAM?
  - How to detect a collision between ball and the paddle?

Report:

No report is needed for Exercise #1. Report for Exercise #2 should give answers to the above four questions.

These two exercises should be completed in one week and move on to the project.