Overview and Background

When working with computer hardware, data often has to be moved from one location to another. This can be between two different chips or between a computer and a FPGA board. Serial communication is a way to send data one bit at a time, that is you are sending the bits in serial. There are different types of serial communication. Three of the most common are I2C, SPI, and UART.

The purpose of this assignment is to implement a UART on the DE10-Lite board for use of serial communications with your computer via a USB to serial adapter.

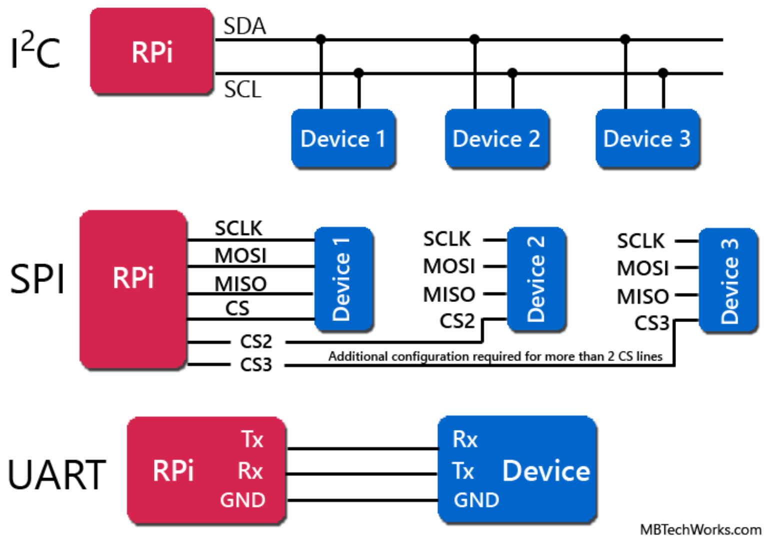


Figure 1, example of communication methods implemented with a Raspberry Pi (RPi)
(Source: https://www.mbtechworks.com/hardware/raspberry-pi-UART-SPI-I2C.html)

A universal asynchronous receiver-transmitter (UART) is a computer hardware device that is used for asynchronous serial communication. Asynchronous serial communication does not have a dedicated clock signal and instead uses start and stop bits to indicate data messages. Data messages are sent at a specific rate known as the baud rate, that is the number of bits per second. UART uses 2 pins for communication, Tx and Rx, and shares a common GND. (Tx: Transmitter. Rx: Receiver).

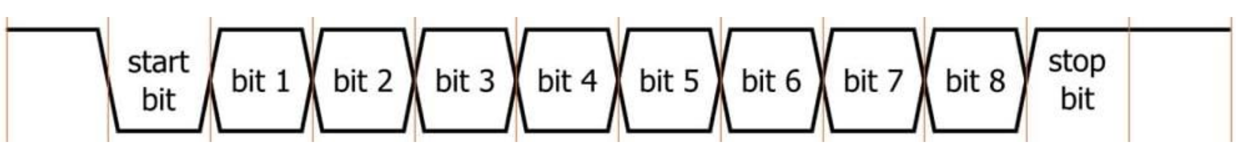


Figure 2, UART bit sequence example.
(Source: https://www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart/)

USB Adapter and GPIO Pins

Be careful when using the USB adapter and the GPIO pins. Plugging things in incorrectly can damage your FPGA board, the USB adapter, or potentially your computer.

The USB adapter has jumpers to ensure that it is operating at the correct voltage level, make sure that the pins marked 3v3 are the pins that are jumpered together. The USB adapter *should* be in the package in this configuration so you should *not* have to change anything, but double check.

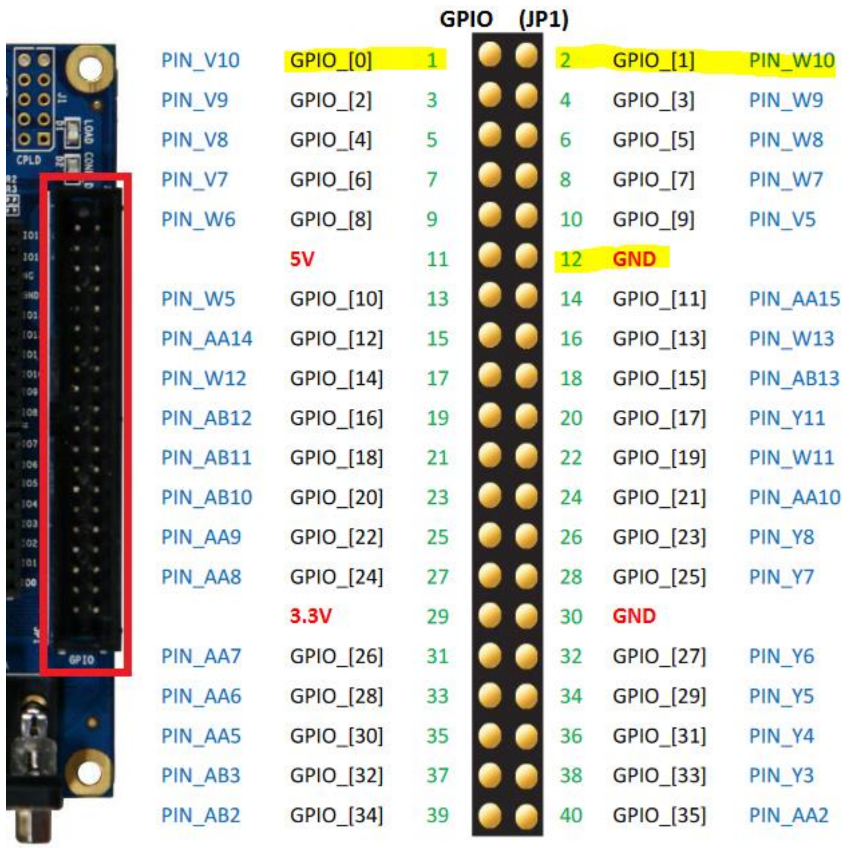


Figure 3, GPIO Pins on DE10-Lite. (Source: DE10-Lite_User_Manual.pdf on Canvas)

We will be using GPIO[0], GPIO[1], and GND. These pins are highlighted in Figure 3.

Assignment

Implement a UART with baud rate 9600 on your DE10-Lite board and interface to your computer via PuTTY (more information about PuTTY provided in later pages). This UART should be capable of transmitting and receiving characters. The transmitted characters will originate from the onboard switches while the received characters will be display on two of the 7-segment displays. The transmitted characters should be shown in hexadecimal, you do not need to decode the ASCII characters.

- Use SW[7:0] to load your 8-bits to transmit. Display the hex value of these bits on HEX1 and HEX0.
- Display a hex value of the received data bits from your computer on HEX5 and HEX4.
- Use KEY0 as a synchronous reset.
- When you press KEY1 it should transmit one group of data bits. That is, the start bit, the 8 data bits, and the stop bit.
- Use GPIO Pin 0 as a Tx pin from your FPGA device. This would then be plugged into the Rx pin of your USB adapter.
- Use GPIO Pin 1 as a Rx pin for your FPGA device. This would then be plugged into the Tx pin of your USB adapter.

Write a testbench to test the transmit section of your UART module.

Use an ASCII table to verify that your UART can transmit and receive as expected while interfaced with PuTTY. For example, the capital letter A has a hexadecimal value of 8'h41.

The recommended architecture is shown below, implement baud rate to send/receive the bits at the appropriate rate with a delay feature similar to the counters used in previous labs (hint: count down instead of up).

Make sure that you do not introduce a clock other than the FPGA's CLOCK_50 to the design as this can cause timing issues and result in metastability.

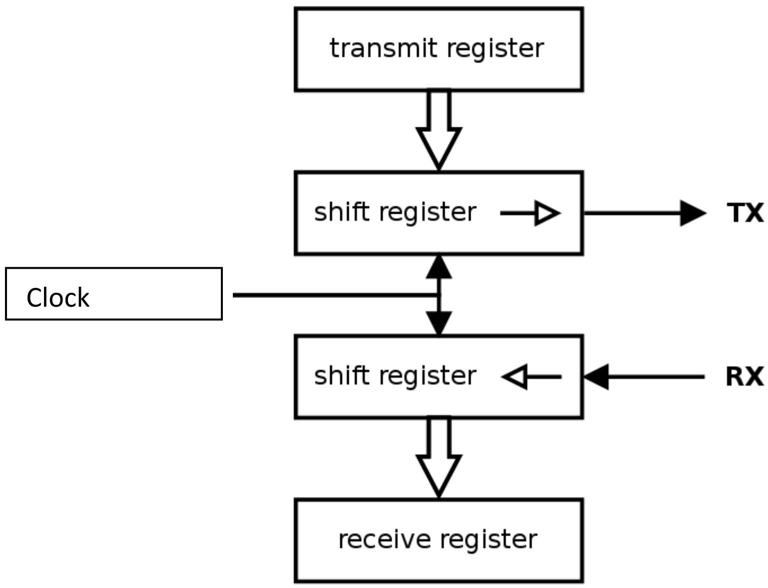


Figure 4, example UART architecture. (Source: https://en.wikipedia.org/wiki/File:UART_block_diagram.svg)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Figure 5, ASCII Table. (Source: https://www.asciitable.com/)

Deliverables

Submit the following to Canvas:

- Your Verilog code for the UART
- Your Verilog code for the testbench
- A screenshot showing a portion of the ModelSim simulation that you used to verify the transmit portion of your code
- A video displaying your FPGA board after receiving: Hello FPGA!
- A screenshot of your PuTTY window after sending: Hello!

Late Policy:

Less than 1 Day late: -5/100 points deducted
Between 1 and 2 Days late: -10/100 points deducted
Between 2 and 3 Days late: -20/100 points deducted
Between 3 and 4 Days late: -40/100 points deducted
More than 4 Days late: Not accepted

Useful Information

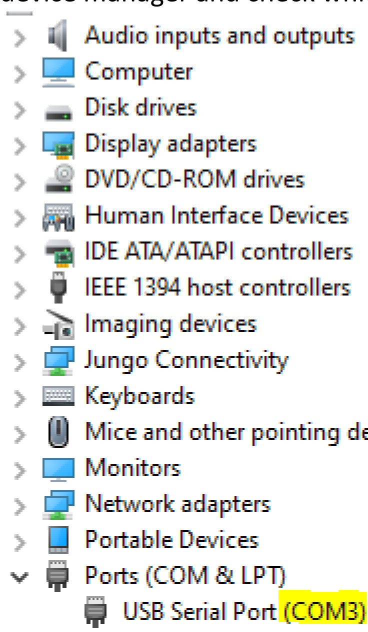
- DE10-Lite User Manual: on Canvas
    - Refer to section 3.5 for information on GPIO pins
- Basics of UART Communication: https://www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart/

- Additional UART information: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
- UART Baud rate information: https://www.allaboutcircuits.com/technical-articles/the-uart-baud-rate-clock-how-accurate-does-it-need-to-be/
- UART From Scratch on a breadboard: https://www.youtube.com/watch?v=aE5VTp_eMN4
- ASCII: https://en.wikipedia.org/wiki/ASCII

Guide to using PuTTY and the USB UART adapter

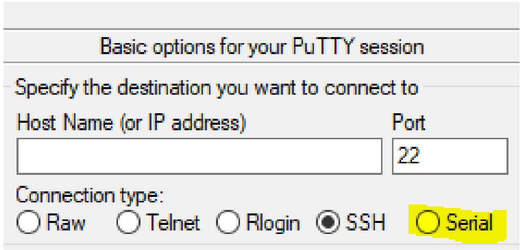
Important note: If you have an active PuTTY window open, you will not be able to upload new code to your FPGA board. Close the PuTTY window and if that doesn't work, unplug the USB UART device. This issue is related to the USB drivers for the adapter and the programmer.

- Use the one of the following guides that is appropriate for your device to install PuTTY:
  - Windows: https://www.ssh.com/academy/ssh/putty/windows/install
  - Mac: https://www.ssh.com/academy/ssh/putty/mac
  - Linux: https://www.ssh.com/academy/ssh/putty/linux
- Plug in the USB UART adapter
- Finding the serial port for the USB device
  - Windows
    - Open up device manager and check which serial port the USB device is listed as

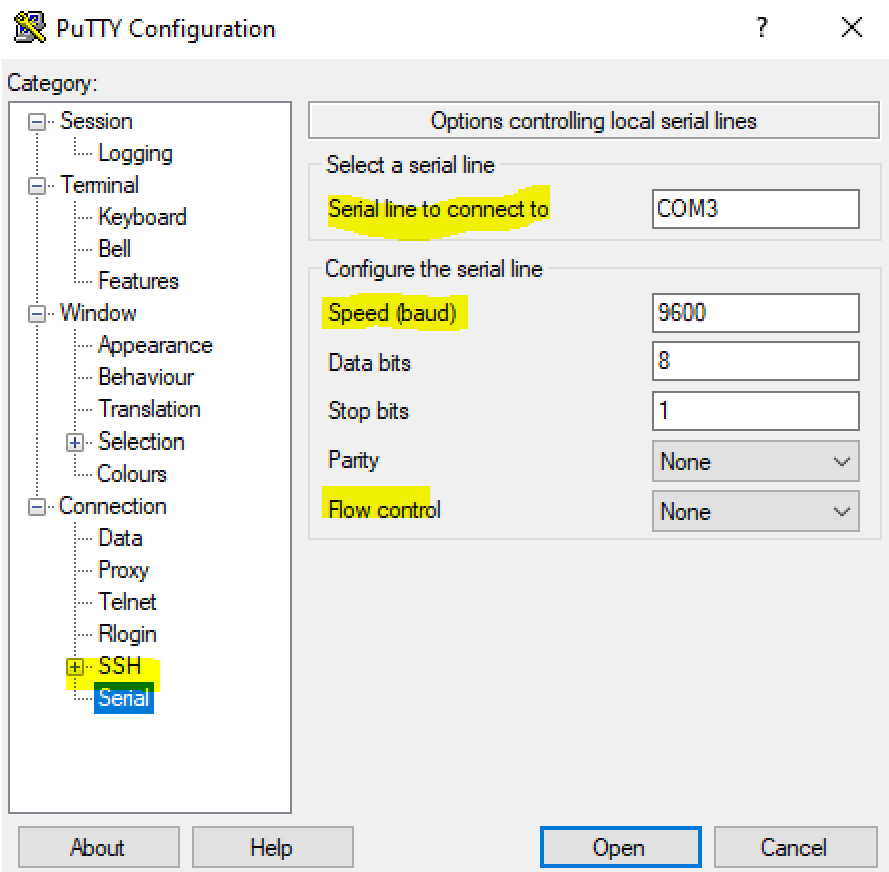


  - Mac
    - Open terminal and type: ls /dev/*
    - Note the port number listed for /dev/tty.usbmodem* or /dev/tty.usbserial*
    - The port number is represented with * here.
  - Linux
    - Open terminal and type: ls /dev/tty*
    - Note the port number listed for /dev/ttyUSB* or /dev/ttyACM*
    - The port number is represented with * here.

- Open PuTTY and change the session from SSH to Serial

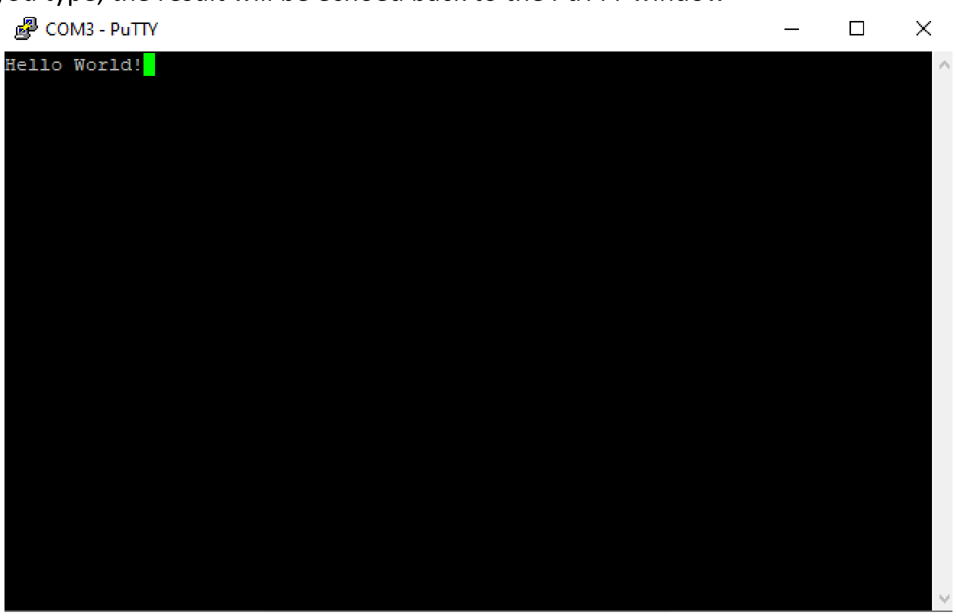


-
- Change to the serial tab and perform the following changes so that your window matches the screenshot (your serial port may be different)
  - Change the serial line to the correct one
  - Change the speed (baud) to 9600
  - Turn off flow control

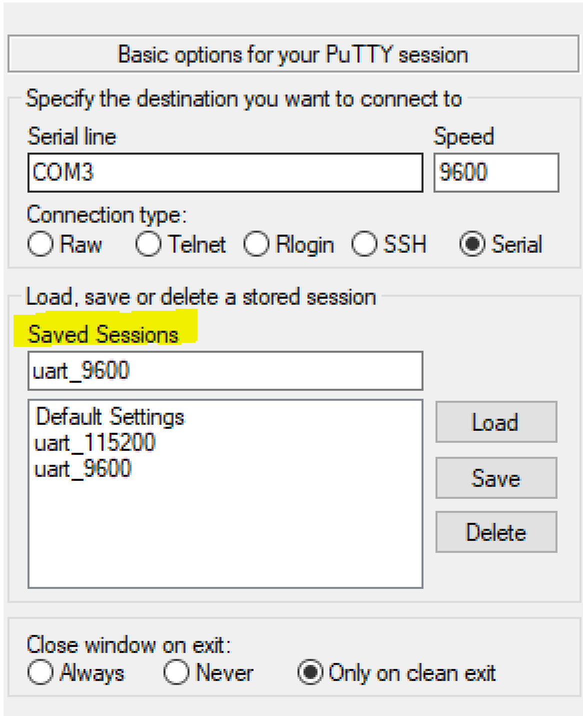


  -
- Click open and PuTTY will open a new window like the blank one below

- 

- Currently, if you type nothing with happen in the PuTTY window
- Connect the RX and the TX pins on the USB device together with a jumper wire
- Now if you type, the result will be echoed back to the PuTTY window
  - 

- You can save these settings to load later:

- 
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
COM3
Speed
9600
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
uart_9600
Default Settings
uart_115200
uart_9600
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit