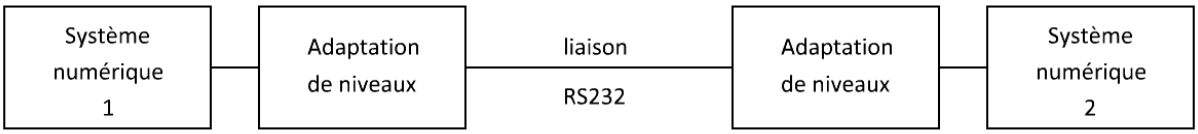
SYNTHESE LOGIQUE

UART (Universal Asynchronous Receiver Transmitter)

1. Présentation

La liaison série permet la communication entre deux systèmes numériques en limitant le nombre de fils. La liaison appelée communément « port série » dans les PC est normalisée par le standard recommandé RS232. Le standard RS232 est utilisé dans l'industrie pour relier des équipements tels que les automates, les appareils de mesure, les machines-outils, … , pour sa simplicité de programmation et sa robustesse.

Les bits de données sont transmis en série à une certaine vitesse exprimée en bauds (bits par seconde). La transmission série nécessite au minimum trois fils : une masse de référence, un fil (TX) pour la transmission de données et un troisième (RX) pour la réception des données.



2. Caractéristiques électriques

Les tensions appliquées à la ligne sont référencées par rapport à une masse (0 volt). Elles sont définies comme suit :

Tension	Etat
-Vmax à -3V	1 logique
-3V à +3V	interdit
3V à Vmax	0 logique

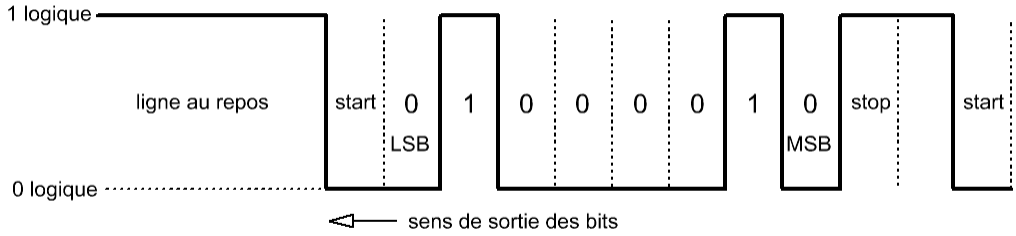
La tension $V_{max} = 12\ V$ est la plus rencontrée. L'adaptation des niveaux électriques se fait avec des circuits spécialisés (ex : le MAX232).

Avant adaptation	Après
Niveau logique 0 = 0V	Niveau logique 0 = +12V
Niveau logique 1 = +5V	Niveau logique 1 = -12V

3. Protocole de transmission

Pour que la transmission se fasse correctement, il est nécessaire que les deux équipements utilisent les mêmes règles, c'est le protocole de transmission. Les paramètres entrant en jeu sont les suivants :

- *Longueur des mots* : 7 ou 8 bits
- *Vitesse de transmission* : réglable à partir de 110 bauds (110, 300, 1200, 2400, 4800, 9600, 19200 …).
- *Parité* : On ajoute un bit à la donnée de telle sorte que le nombre de tous les bits à 1 soit pair (en parité paire) ou impair (en parité impaire).
- *Bit de start* : La ligne au repos est à l'état logique 1. Pour indiquer un début de transmission, elle passe à 0 (start).
- *Bit de stop* : après la transmission, la ligne est positionnée au 1 logique pendant une à deux périodes en fonction du nombre de bits de stop.

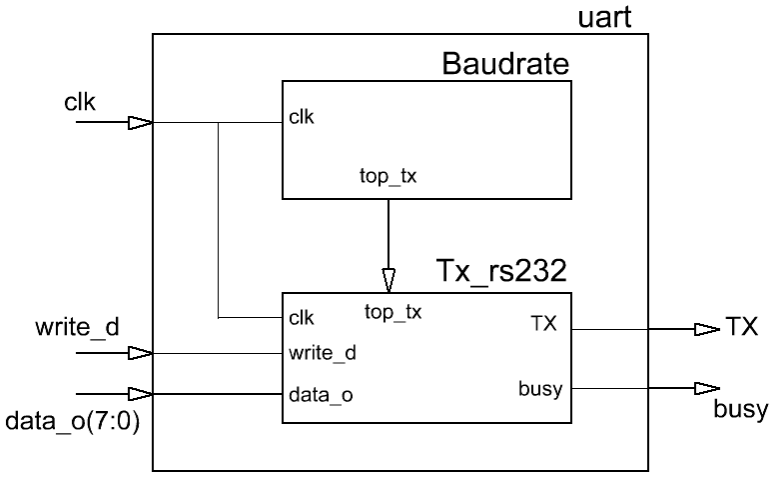


4. <u>But du projet</u>

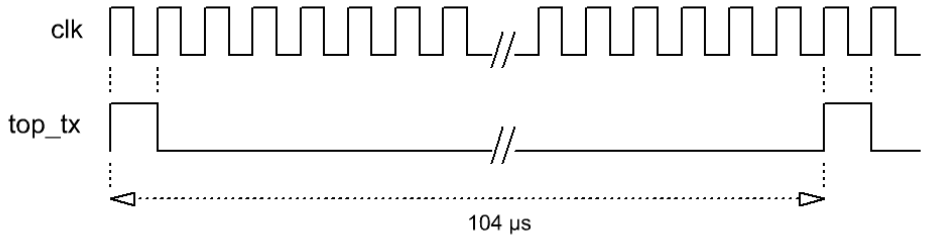
Établir une liaison série asynchrone entre un composant FPGA et un PC en adoptant le protocole suivant :

- *<u>Longueur des mots</u>* : 8 bits
- *<u>Vitesse de transmission</u>* : 9600 bauds
- *<u>Parité</u>* : non
- *<u>Bit de start</u>* = 0
- *<u>Un bit de stop</u>* = 1

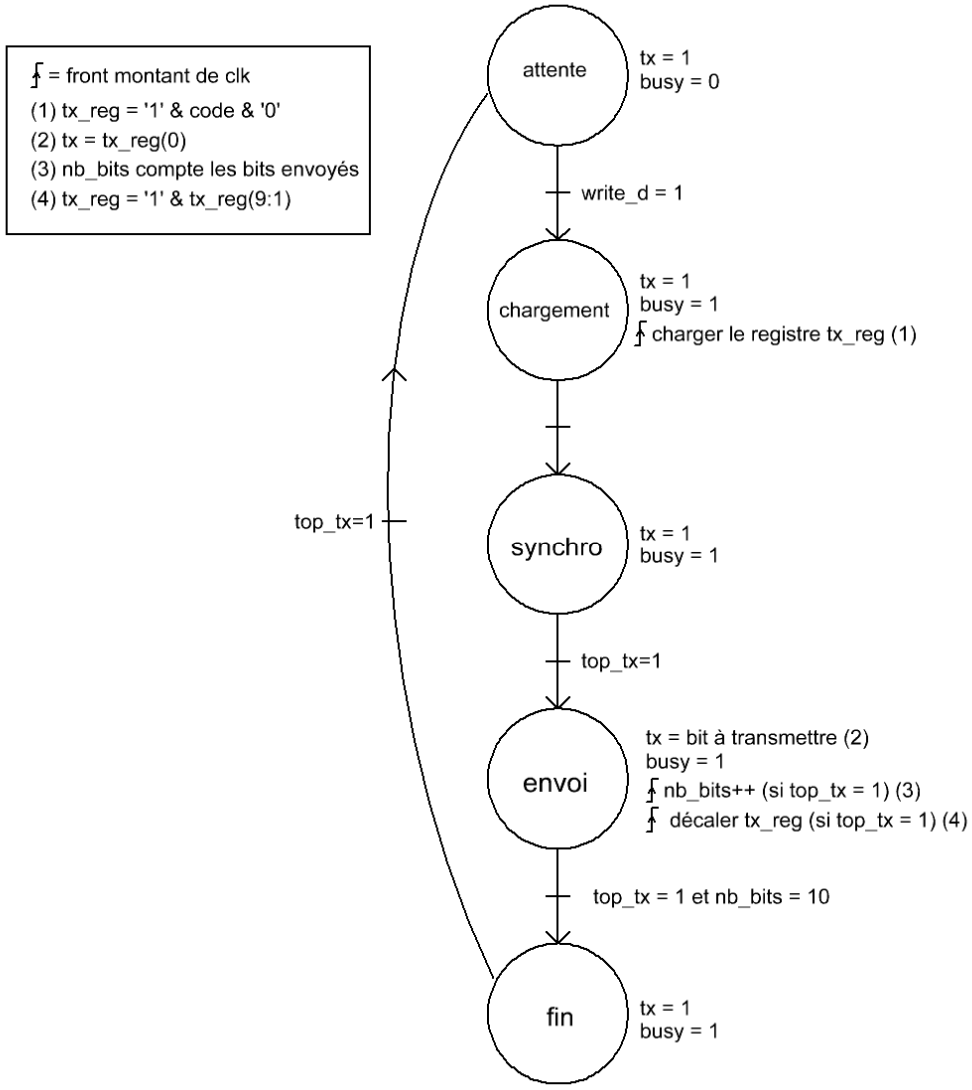
Le but du TP est d'établir une transmission de données de la maquette vers le PC. L'UART à réaliser ne s'occupe pas de la transmission de données vers le PC, il est représenté par le schéma suivant :



☑ **Baudrate** : fabrique un signal impulsionnel **top_tx** de fréquence 9600 Hz à partir de l'horloge 100 MHz.
Le signal **top_tx** reste à 1 pendant une période d'horloge.



☑ **Tx_rs232** : si **write_d** passe à 1, l'octet **data_o** est transmis à la fréquence **top_tx** sur la ligne. Le signal **busy** indique que la transmission n'est pas finie. La machine d'états suivante décrit le fonctionnement du module **tx_rs232**. La condition **(clk)** signifie que l'action se déroule sur front montant d'horloge. La condition **(clk et top_tx)** signifie que l'action est exécutée sur front montant si **top_tx** est à 1.



5. Travail à effectuer

Partie 1 : cette première partie consiste à envoyer au PC le code ASCII sélectionné sur les Switchs (7 :0). L'utilitaire **Teraterm** affichera sur l'écran du PC les caractères transmis.

- Écrire les programmes des blocs **baudrate** et **Tx_rs232** puis les simuler.
- Associer ces 2 modules dans un bloc nommé **uart**. Simuler son fonctionnement.
- Ajouter un fichier de contraintes au projet pour positionner les signaux sur les broches du composant. **Data_**o sur les switchs, **write_**d sur BtnC, busy sur led(0). Compiler le projet et configurer le composant pour tester le fonctionnement sur la maquette.

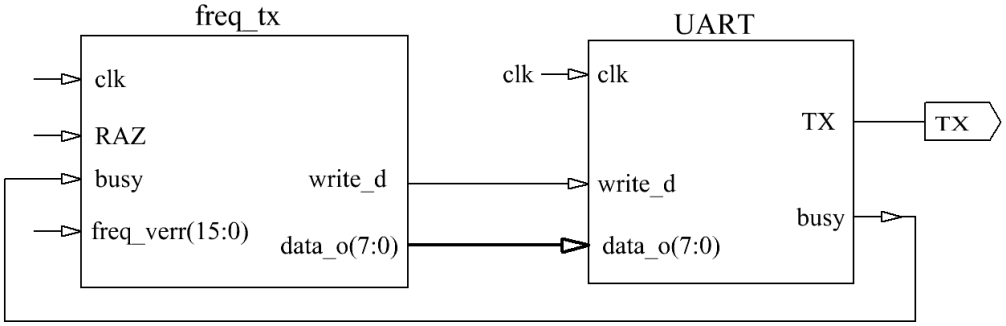
Partie2 : dans cette deuxième partie, nous allons doter notre fréquencemètre d'un bloc **uart** qui permettra d'envoyer la fréquence mesurée au PC.

L'affichage sur l'écran du PC se fera sous cette forme : chaque fréquence mesurée est affichée sur 4 caractères suivis des lettres 'H' et 'z', il y a un retour en début de ligne suivante après chaque fréquence affichée.

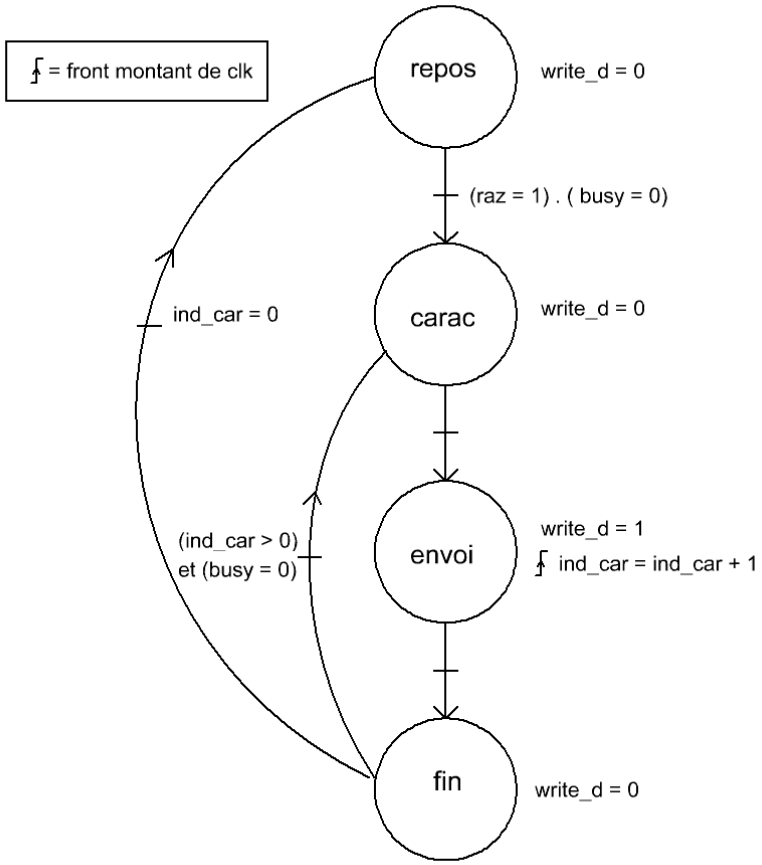
1200Hz
1500Hz
2000Hz

Type d'affichage à l'écran

- Faire une copie du répertoire **freq**, le nommer **freq_uart**
- Ouvrir le nouveau projet **freq_uart**, lui ajouter les fichiers **UART, Baudrate et Tx_rs232**.
- En fait, le fréquencemètre du TP1 va s'enrichir de 2 nouveaux composants : l'**UART** qui est déjà validé et une machine d'états **freq_tx** qui sera créée. Le schéma suivant montre comment sont connectés ces 2 nouveaux composants (voir le synoptique complet en dernière page).



- Le fonctionnement de **freq_tx** est décrit par la machine d'états suivante :



- Ecrire le programme **freq_tx.vhd** en déclarant un signal **ind_car** de taille 3 bits. L'aiguillage des données vers la sortie **data_o** se fera comme suit :

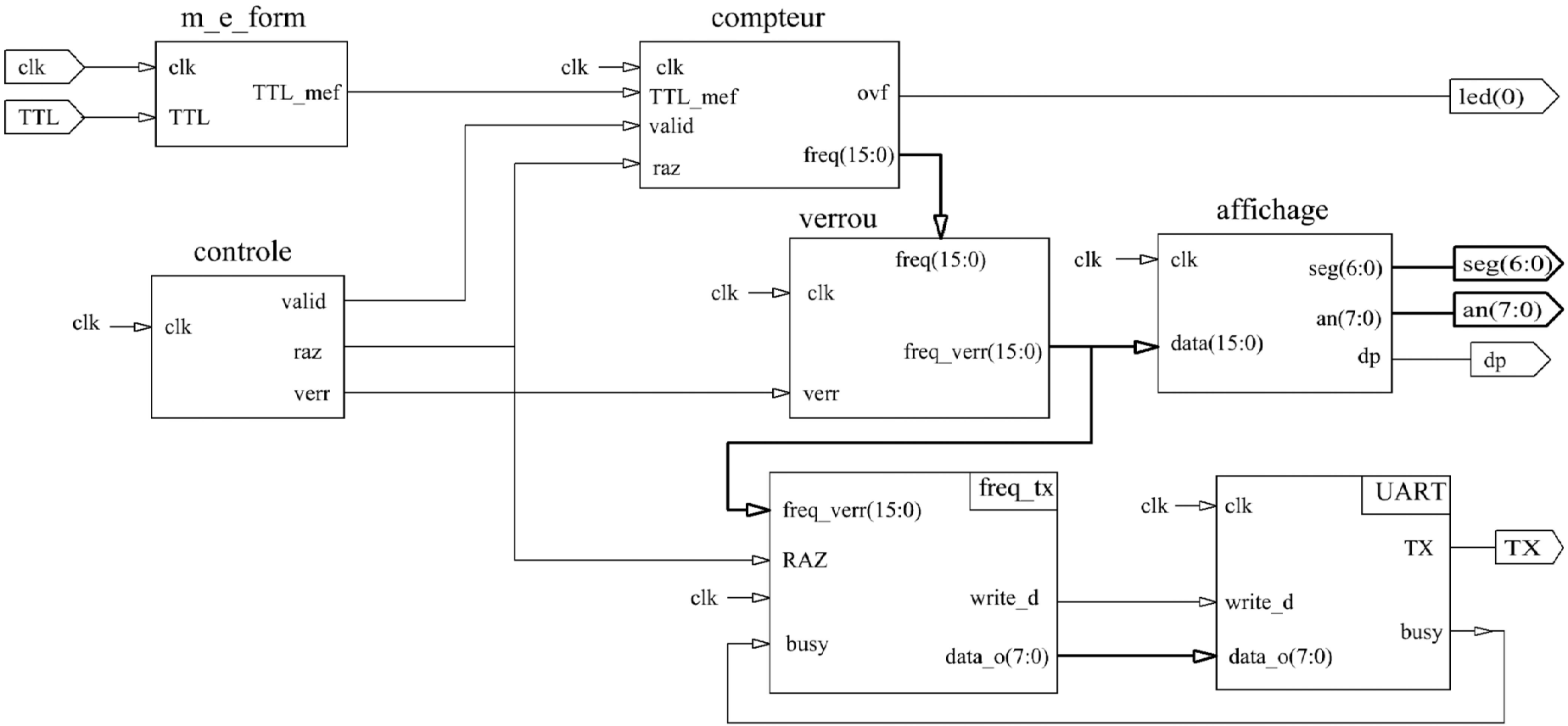
```
data_o <= ("0011" & freq_verr(15 downto 12)) when ind_car="001" else
       ("0011" & freq_verr(11 downto 8)) when ind_car="010" else
       ("0011" & freq_verr(7 downto 4)) when ind_car="011" else
       ("0011" & freq_verr(3 downto 0)) when ind_car="100" else
       "01001000" when ind_car="101" else -- H
       "01111010" when ind_car="110" else -- z
        "00001010" when ind_car="111" else -- LF (Line Feed)
       "00001101";   -- CR (carriage return)

end Behavioral;
```

- Modifier le programme de niveau 1 en ajoutant les nouveaux composants.
- Compléter le fichier de contraintes en conséquence.
- Compiler le projet et configurer le composant pour tester le fonctionnement du fréquencemètre avec affichage de la fréquence sur la maquette et sur le PC.

ASCII Code

Row Number \ Column Number	000	001	010	011	100	101	110	111
0000	*NUL*	*DLE*	◊	0	@	P	`	p
0001	*SOH*	*DC1*	!	1	A	Q	a	q
0010	*STX*	*DC2*	"	2	B	R	b	r
0011	*ETX*	*DC3*	#	3	C	S	c	s
0100	*EOT*	*DC4*	$	4	D	T	d	t
0101	*ENQ*	*NAK*	%	5	E	U	e	u
0110	*ACK*	*SYN*	&	6	F	V	f	v
0111	*BELL*	*ETB*	'	7	G	W	g	w
1000	*BS*	*CAN*	(	8	H	X	h	x
1001	*HT*	*EM*	)	9	I	Y	i	y
1010	*LF*	*SUB*	*	:	J	Z	j	z
1011	*VT*	*ESC*	+	;	K	[	k	{
1100	*FF*	*FS*	,	<	L	\	l	|
1101	*CR*	*GS*	-	=	M	]	m	}
1110	*SO*	*RS*	.	>	N	^	n	~
1111	*SI*	*US*	/	?	O	_	o	*DEL*


m_e_form
clk
TTL
clk
TTL
TTL_mef
compteur
clk
clk
TTL_mef
valid
raz
ovf
freq(15:0)
led(0)
controle
clk
clk
valid
raz
verr
verrou
clk
freq(15:0)
clk
freq_verr(15:0)
verr
affichage
clk
clk
data(15:0)
seg(6:0)
an(7:0)
dp
seg(6:0)
an(7:0)
dp
freq_tx
freq_verr(15:0)
RAZ
clk
busy
write_d
data_o(7:0)
UART
clk
clk
write_d
data_o(7:0)
TX
busy
TX