Communication via une liaison série

UART: Universal asynchronous receiver/transmitter

Emetteur/Récepteur asynchrone universel

Mandat

Le but de ce projet est de concevoir et réaliser une communication via une liaison série avec une seconde carte connectée à la plateforme REPTAR. Le projet comprendra la réalisation d'un UART, l'interfaçage de ce bloc puis la réalisation d'un programme d'application afin de tester la communication.

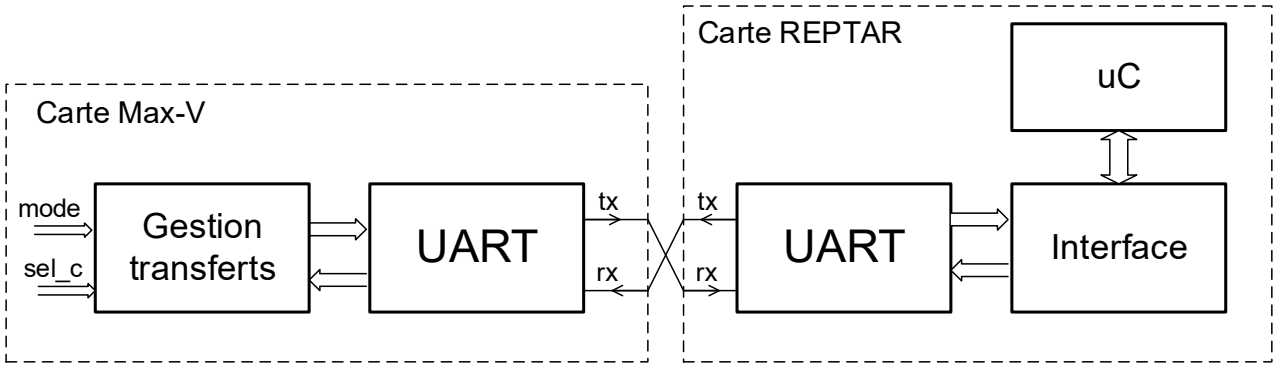
La première partie du projet est de concevoir et réaliser un émetteur asynchrone universel. Le récepteur est fourni dans le projet Logisim. En anglais, ce module est nommé : **UART** pour **Universal asynchronous receiver/transmitt**er. Il s'agit du principe le plus simple de liaison série. Cette liaison série fut l'une des premières utilisées pour communiquer avec différents périphériques à l'aide de 2 fils uniquement, tel que le clavier, un terminal (écran), etc. Le fonctionnement de base, soit la transmission série des bits d'information, reste actuel. Toutes les transmissions à haut débit récente utilisent ce principe de transfert en série. Nous pouvons citer le bus USB, dont la version 3.0 est à 5Gigabits/sec et la version 3.1 à 10Gigabits/sec. Il y a aussi le bus PCI-express qui utilise une fréquence de 8Gigabits/sec pour la version 3.x et 16Gigabits/sec pour la version 4.0 par lien. Pour augmenter le débit de transfert entre deux équipements le PCI-express permet d'avoir plusieurs liens en parallèle, soit 1, 2, 4, 8, 16 ou 32.

Vous avez de plus ample information sur Wikipedia: https://fr.wikipedia.org/wiki/UART

La seconde partie comprend la conception et la réalisation d'une interface, pour le module UART RX/TX, avec le bus du processeur de la carte REPTAR. Cette partie comprendra la spécification de la gestion de flux avec le module UART et la définition d'un plan d'adressage.

Finalement, un programme C sera réalisé afin de pouvoir valider l'ensemble par une application simple utilisant la communication série. Une carte Max-V sera utilisée pour disposer d'un second module UART RX/TX en vis-à-vis de celui développé dans la carte REPTAR.

Structure générale du système :

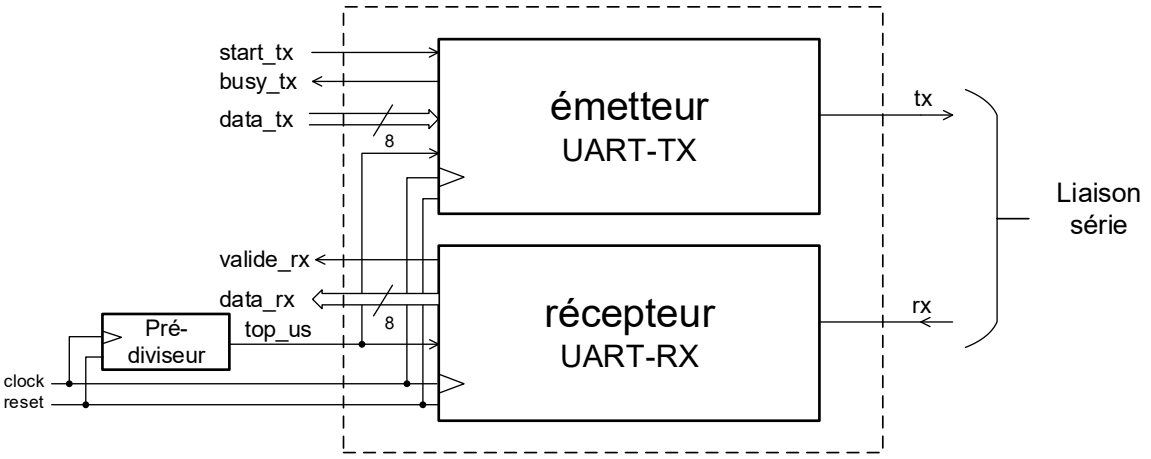


Le module de test est intégré dans une carte Max-V. Une carte déjà programmée vous sera fournie. Les fonctionnalités de ce module sont données en annexe.

La connexion entre le module de test et la carte REPTAR est faite par le connecteur 80pôles.

1ère partie : émetteur série

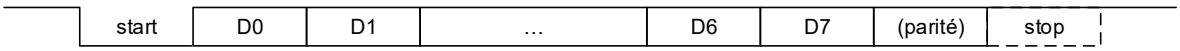
Voici la structure du module UART comprenant les blocs émetteur et récepteur:



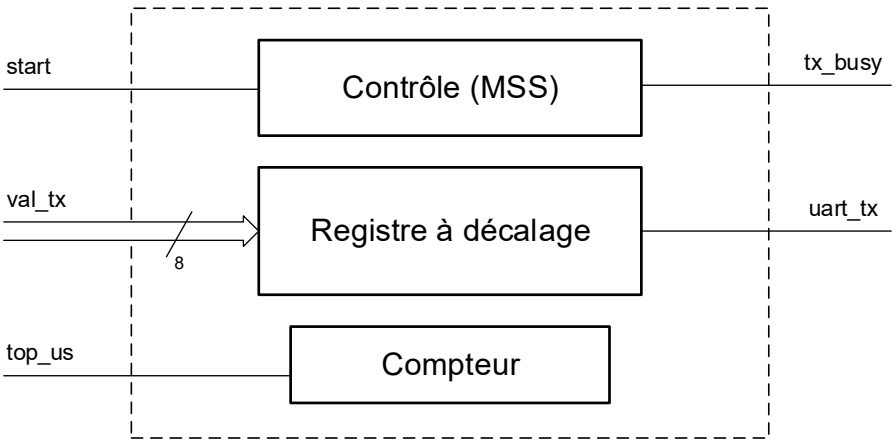
Vous devez réaliser le bloc émetteur. Les autres blocs, soit le récepteur et le pré-diviseur, sont fourni dans le projet Logisim.

Description des trames UART :

La suite de bits envoyés en série est nommée trame. Le start bit est envoyé en premier, il est à l'état '0'. Puis les bits de la donnée sont envoyés en série en commençant par les bits de poids faible (D0, D1, …). A la suite des bits de la donnée, le bit de parité est transmis. Il est facultatif (non demandé dans ce labo). Puis la ligne se met dans l'état de repos, soit à l'état '1'. Il y a au minimum un bit, celui-ci est nommé le stop bit. Voici le chronogramme de fonctionnement de la ligne série :



Voici le schéma bloc de l'émetteur série UART-TX :



Spécification du fonctionnement de l'émetteur UART-TX :

- Après le reset, la sortie série, uart_tx, est à l'état '1' et le signal tx_busy est à '0'
- Lorsque le signal start est actif (état '1') la donnée val_tx est mémorisée, puis la transmission série est démarrée. La transmission commence par les bits de poids faible
- Chaque bit doit être transmis pendant 1/8 µs, soit une division par 8 de signal top_us.
- A la fin de la transmission, la sortie uart_tx doit être à l'état haut ('1') et la sortie tx_busy est désactivée ('0').

Vous disposez d'un pré-diviseur par 200, qui génère un top_us à une fréquence de 1MHz, soit une période de 1µs.

Référence : Donnée du labo de l'unité ARO1 « laboratoire_ARO1_UART_17.pdf »

2ème partie : interface du module UART

Cette partie comprend la réalisation d'une interface du module UART RX/TX.

Le projet Logisim fourni comprend déjà quelques entrées/sorties qui sont indiqués dans le plan d'adressage ci-dessous. Celui-ci prévoit le décodage d'une zone de 16 adresses de mots de 16 bits.

Adresse CPU	Read	Write
	D15 0	D15 0
0x1900 0000	Cst (0x6543, test)	not used
0x1900 0002	"0000 0000" switch7..0	not used
0x1900 0004	*"0000 0000" Leds7…0* **not availlable**	*"0000 0000" Leds7…0* **not availlable**
0x1900 0006	aff_7seg2 aff_7seg1	aff_7seg2 aff_7seg1
0x1900 0008		
0x1900 000A		
…		
0x1900 001E		

Dans le projet fourni, les leds ont été utilisées pour afficher la donnée reçue par le bloc RX. Elles ne sont plus utilisables pour votre projet.

Spécification des fonctionnalités de l'interface du module UART:

- Doit permettre l'écriture d'une donnée data_tx à transmettre
- Proposer une action pour le démarrage du transfert en série (émission : TX)
- Doit permettre la lecture de la donnée reçue en série, soit data_rx.
- Doit gérer 2 bits de statut qui doivent pouvoir être lu par le CPU, soit :
  - new_data : indique qu'une nouvelle donnée a été reçue par Uart_recept.
  - error : ce bit est activé si une seconde donnée est reçue alors que la précédente n'a pas été lue (bit de statut new_data actif)
- Proposer une gestion des flags new_data et error pour leur remise à '0' après une lecture de la données série reçue (data_rx). Vous expliquerez votre choix sur l'instant pour quittancer ces bits de statut.

3ème partie : réalisation d'une application

Cette partie comprendra la réalisation d'un programme C afin de pouvoir valider l'ensemble par une application simple avec la communication série réalisée et son interface. Une carte Max-V sera utilisée pour disposer d'un second module UART RX/TX en vis-à-vis de celui développé dans la carte REPTAR. Voir la spécification de la carte Max-V en annexe.

Spécification de l'application pour la mise en œuvre et le test de l'UART:

- Un premier programme doit permettre de lire les caractères envoyés par le design de la carte Max-V et de les afficher dans la console (printf).
  - choisir design Max-V en « mode envoie d'un caractère »
  - puis choisir le « mode envoie d'une chaine de caractères »
    - Est-ce que la chaine affichée est complète ?
    - Y a-t-il une ou plusieurs erreurs lors de la réception ?
- Une seconde version doit permettre d'envoyer un caractère, puis de le réceptionner et de l'afficher dans la console (printf).
  - configurer le design Max-V en « mode écho simple »
  - puis vous testerez le « mode écho avec conversion »
- Application finale : acquérir une chaine de caractères (tstc, getc) puis l'envoyer à votre UART et afficher les caractères reçus en retour dans la console (printf).
  - Vérifier votre application en « mode écho simple » et en « mode écho avec conversion »

Référence: APIs pour u-boot, voir document "u-boot-api.pdf"

Notes : Afin que votre système soit dans son état initiale pour commencer vos tests, respecter les étapes suivantes :

- Connecter la carte Reptar à la carte maxV par le connecteur 80p, puis alimenter les cartes.
- Programmer la Spartan 6 de la carte Reptar.
- Appuyer sur le bouton reset de la carte Reptar (SP3/SP6).
- Appuyer sur le bouton reset de la carte Max-V (SW9 rouge)

A rendre :

Dans un **SEUL** fichier PDF :

- Rapport avec toutes les explications sur les étapes demandées dans l'énoncé (1ère , 2ème et 3ème parties) ainsi que la copie de tous les documents nécessaires pour la correction (schéma Logisim, programme C, etc.)

Dans **une archive** de type *.zip ou *.tar.gz :

- Un fichier zip/tar avec les fichiers de votre répertoire de travail
  - soit les fichiers *.circ, *.c, et autres fichiers modifiés
  - **sans** le répertoire "include" (il fait 3.3MB!). Merci

Marche à suivre

1ère partie :

1) Compléter le bloc Uart_emet du projet Logisim afin de concevoir et réaliser un émetteur série.
2) Simuler le bloc Uart_emet afin d'en vérifier le fonctionnement (rendre les chronogrammes).

2ème partie :

3) Etudier et proposer une gestion de flux pour l'interface du module UART pour répondre aux fonctionnalités de mandées de l'interface. Vous compléterez le plan d'adressage correspondant à votre solution.
4) Compléter le bloc lba_usr_interface fourni pour implémenter tous les éléments nécessaire à votre gestion de flux défini au point précédent.
5) Simuler votre composant lba_usr_interface dans Logisim afin d'en vérifier le fonctionnement (rendre les chronogrammes).
6) Vérifier le fonctionnement de votre interface à l'aide de commandes u-boot (md et mw). Vérifier le fonctionnement de votre gestion de flux pour l'envoi et la réception de donnée en série.

   Pour ces tests vous utiliserez la carte Max-V connectée sur le connecteur 80 pôles.

3ème partie :

7) Compléter le programme C fourni pour les différentes propositions de variantes d'applications.
8) Tester le système, pour chaque version du programme, avec la carte Max-V connectée sur le connecteur 80 pôles.
9) Vous devez faire valider votre montage final par votre professeur

Annexe

Spécification du design de la carte de test Max-V :

Vous disposez d'un design de test intégré dans une carte Max-V. Celui-ci comprend un module UART avec RX et TX. Celui-ci dispose de plusieurs modes de fonctionnement pour le module d'émission (canal TX).

Voici la Spécification des modes de fonctionnement de l'émetteur du design de test :

mode(1..0)	Description du fonctionnement de l'émetteur du design de test de la carte Max-V
00	Mode écho simple (caractère reçu est renvoyé)
01	Mode écho avec conversion les lettres minuscules converties en majuscule et vice-versa. autres caractères pas modifiés
10	Envoie d'un caractère lors d'une activation du bouton start (SW1) Possibilité de sélectionner le caractère d'une table comprenant 8 valeurs. Voir ci-dessous.
11	Envoie d'une chaine de caractères, soit: "Hello world IFS 2018" le message est envoyé à chaque start (entrée à définir)

Indications des dip-switch utilisés pour le choix du mode du design de test :

- mode(1..0) dip-switch 1 – 0

Description du fonctionnement du mode d'envoi d'un caractère, soit mode = "10" :

- start bouton SW1, commande l'envoi du caractère sélectionné.
- sel_car dip-switch 4 - 3 – 2, permet de sélectionner le caractère envoyé

sel_car(2..0)	Caractère envoyé
000	0x42, caractère ASCII 'B'
001	0x6F, caractère ASCII 'o'
010	0x6E, caractère ASCII 'n'
011	0x6A, caractère ASCII 'j'
100	0x6F, caractère ASCII 'o'
101	0x75, caractère ASCII 'u'
110	0x72, caractère ASCII 'r'
111	0x21, caractère ASCII '!'