6GEI420 – Systèmes Digitaux

Laboratoire #7

Conception de machines à états finis

Hiver 2018

1. Objectifs

- Apprendre à concevoir des machines à états avec Quartus Prime
- Se familiariser avec le protocole de communication UART

2. Méthodologie

Dans ce laboratoire, il sera question de concevoir des circuits séquentiels. Pour ce faire, nous allons utiliser l'outil dans Quartus Prime qui permet de spécifier un design en termes de diagrammes d'états et de le convertir automatiquement en code VHDL qui peut ensuite être utilisé dans un plus gros système. Dans la première partie du laboratoire, vous aurez à suivre les étapes d'un tutoriel pour concevoir un système qui ressemble à une lumière de circulation. Le système devra allumer la lumière verte pendant un certain temps, puis passer à la lumière jaune et finalement par la lumière rouge pour ensuite tout recommencer. Dans la deuxième partie, vous aurez à concevoir un système qui transmet de l'information du FPGA vers votre ordinateur. Cette communication va se faire avec le protocole UART.

3. Tutoriel

Ouvrez le logiciel Quartus Prime et créez un projet avec le nom *lumiere*. Comme pour les autres laboratoires, utilisez un répertoire temporaire sur le bureau pour que vous puissiez l'effacer par la suite.

Une fois que c'est fait, créez un nouveau fichier de type *State Machine File*.

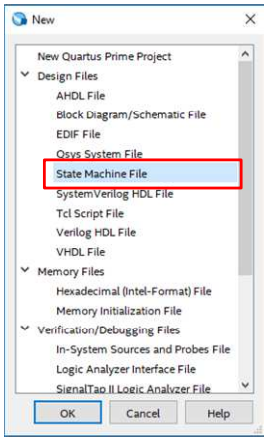

Figure 1. Ouverture d'un fichier de type State Machine File.

Identifiez la section de la fenêtre qui contient les inputs : il devrait déjà y en avoir 2, soient l'horloge et un signal de reset. Nous allons vouloir ajouter une entrée de 3 bits qui va s'appeler *compte*. Ce signal de 3 bits, qui augmentera sa valeur de 1 à chaque seconde (0 à 7), proviendra d'un compteur externe et c'est avec cette valeur de compte qu'on va décider quand la lumière devrait être verte, jaune ou rouge. Pour ce faire, déplacez votre souris dans la région des inputs et cliquez avec le bouton de droite et choisissez *Insert New*. Double-cliquez sur le nom et changez-le pour le nom *compte[2:0]*. Comme certains ont surement pu le deviner, [2:0] indique qu'il y a 3 bits. S'il n'y avait qu'un seul bit, ce serait simplement le nom du signal. Le logiciel vous posera une question et vous n'aurez qu'à cliquer sur OK pour poursuivre. Le logiciel veut simplement nous avertir que, changer le nom d'un signal peut faire en sorte que la machine à états cesse de bien fonctionner (si on l'avait déjà conçu).

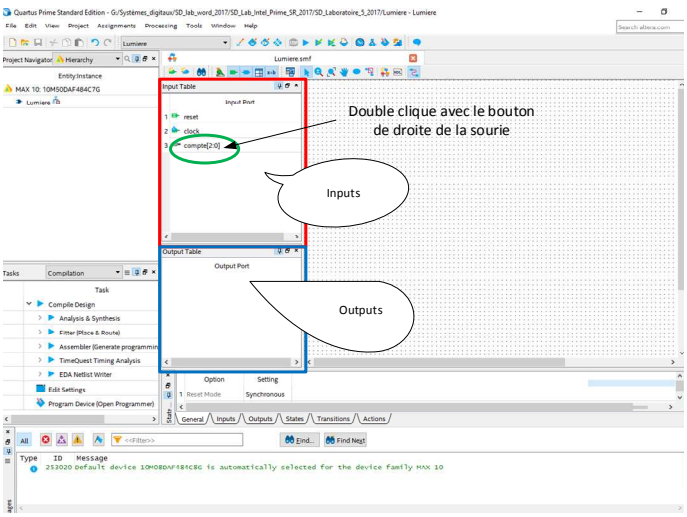

Figure 2. Entrée et sortie.

De la même manière, nous allons vouloir créer 3 sorties : *vert*, *jaune* et *rouge*. Ces signaux seront connectes à des DELs qui vont s'allumer séquentiellement comme des lumières de circulation.

Pour concevoir un système séquentiel, on commence normalement avec une description en langage humain : le système va rester vert pendant 4 secondes, va rester jaune pendant 1 seconde et va rester rouge pendant 3 secondes pour ensuite tout recommencer. Pour la valeur du compte en entrée, on va réutiliser le compteur de 1 seconde du laboratoire précédent. Il va nous donner une valeur qui va de 0 à 7 et qui change à chaque seconde.

Une façon de faire ce système est de se dire que la lumière sera verte quand le compte sera entre 0 et 3. Quand il est égal à 4, la lumière sera jaune. Finalement, quand le compte est entre 5 et 7, la lumière sera rouge.

Voici un croquis du diagramme d'états.

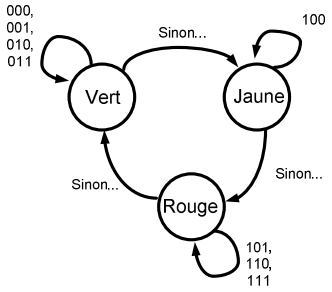

Figure 3 Diagramme d'états

Pour reproduire ce diagramme d'états dans Quartus Prime, nous allons procéder en 3 étapes: ajouter les états, créer les transitions et générer le VHDL.

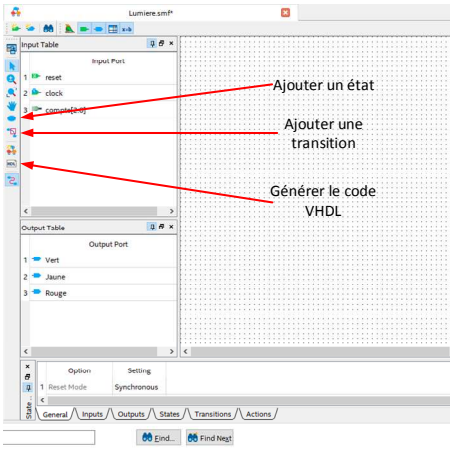

Figure 4. Boutons pour ajouter les états, les transitions et générer le code VHDL.

On commence par ajouter des états. Cliquez sur le rond qui est rose et ajoutez 3 états. Double-cliquez sur les cercles pour changer leurs noms :

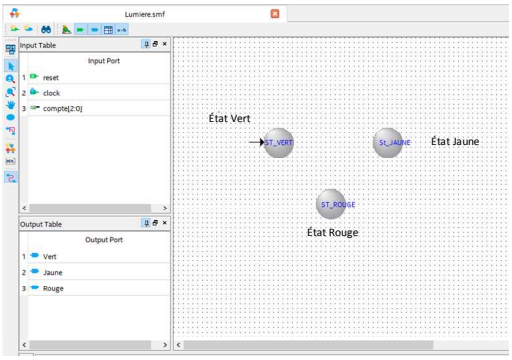

Figure 5. État pour chacune des lumières.

On peut aussi double-cliquer pour spécifier la valeur de la sortie pour chaque état. Pour l'état st_vert, par exemple, on va vouloir que la sortie vert soit 1 et que les sorties jaune et rouge soient 0.

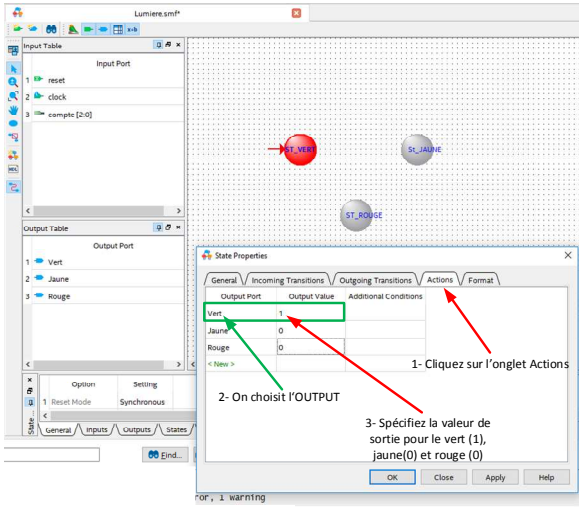

Figure 6. Sortie pour chaque état.

On fait la même chose pour les autres états. Donc, l'état jaune devrait avoir la sortie jaune à 1 et les autres à 0 et l'état rouge devrait avoir la sortie rouge à 1 et les autres à 0.

Une fois terminée, on ajoute les transitions. Pour ce faire, on commence par sélectionner l'outil « transition ». Pour ajouter une transition allant d'un état « vers lui-même », on clique sur l'état lui-même. Pour ajouter une transition d'un état à l'autre, on clique et on garde le bouton appuyé jusqu'à ce qu'on arrive à l'état de destination. Votre diagramme devrait ressembler à ceci :

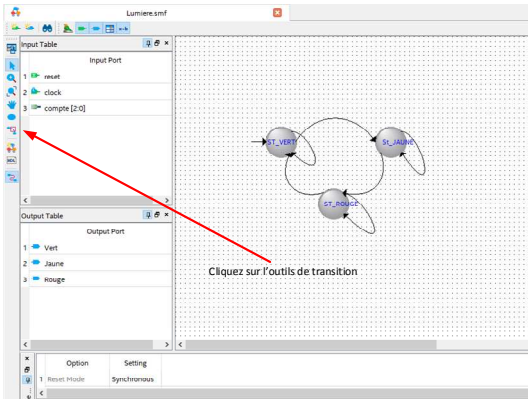

Figure 7. Transition entre états.

On ajoute les conditions pour que ça passe d'un état à l'autre. Pour ce faire, on double clique sur la transition et on écrit la condition. On commence avec la transition de l'état st_vert vers lui-même. La condition est : « tant que le compte est 0, 1, 2 ou 3, je reste ». De façon plus élégante, on pourrait dire « si le compte est plus petit que 4, je reste ». Pour le spécifier, on double clique sur la transition et dans la section Équation, on met la condition *compte* < *4*. La transition allant de st_vert vers st_jaune se produit quand le compte est 4, 5, 6 et/ou 7. Une autre façon de le dire c'est simplement « si la condition de tantôt (compte < 4) n'est pas bonne, prend cette transition pour aller à st_jaune ». Heureusement, le logiciel a une fonction pour ça.

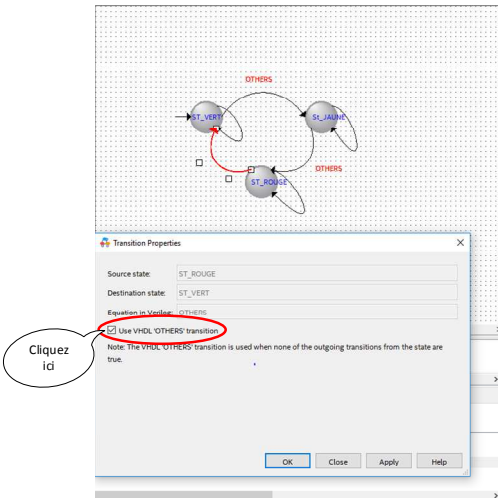

Figure 8. Transition vers un autre état.

Faites la même chose pour st_jaune et st_rouge. Pour st_jaune, compte doit être égal à 4 pour rester (compte == 4). Pour la transition vers l'état st_rouge, cliquez sur le bouton Use VHDL OTHERS. Finalement, pour l'état st_rouge, compte doit être égal à 5, 6 ou 7 pour rester (compte > 4). Sinon, il passe à l'état st_vert.

Finalement, on clique sur le bouton pour générer le code VHDL. Il vous demandera de sauvegarder et par la suite, il vous demandera si vous voulez le code en Verilog, VHDL ou System Verilog : vous choisissez VHDL. Vous attendez quelques secondes et vous aurez une fenêtre qui apparait avec un code tout fait. Si ce n'est pas le cas, vous pouvez appeler le charge de laboratoire pour de l'aide.

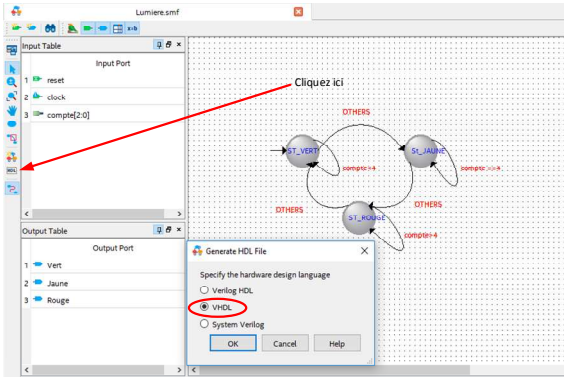

Figure 9. Transformation en code VHDL.

Une fois que vous avez le code VHDL, créez un bloc avec celui-ci. Utilisez-le pour faire un système final avec un compteur et votre système de lumières. Implémentez-le sur FPGA et assurez-vous que ça fonctionne.

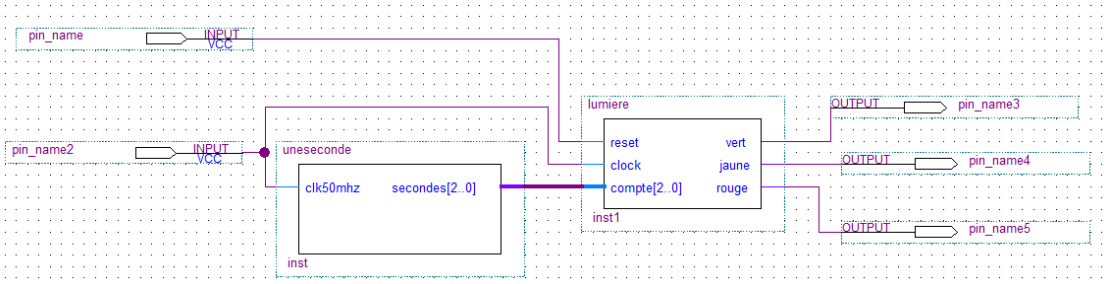

Figure 10. Représentation schématique.

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CLK_10	PIN_N5	10 MHz clock input for ADC (Bank 3B)	3.3-V LVTTL
MAX10_CLK1_50	PIN_P11	50 MHz clock input(Bank 3B)	3.3-V LVTTL
MAX10_CLK2_50	PIN_N14	50 MHz clock input(Bank 3B)	3.3-V LVTTL

Figure 11. Extrait du manuel de l'utilisateur du DE10-Lite

4. Information sur le UART

Dans la deuxième partie du laboratoire, vous aurez à travailler avec une interface UART pour envoyer des données du FPGA vers l'ordinateur. Puisque les ports UART sont moins fréquents sur les ordinateurs modernes, vous utiliserez la puce CP2102 de Silicon Labs pour convertir les signaux USB en UART. Votre ordinateur sera donc connecté à la puce CP2102 par un câble USB et la sortie de cette puce sera connectée à votre FPGA. Quand le pilote de périphérique est installé, l'ordinateur pensera que son port USB est un port UART et il vous permettra de communiquer avec ce port par le biais d'un programme tel que PUTTY.

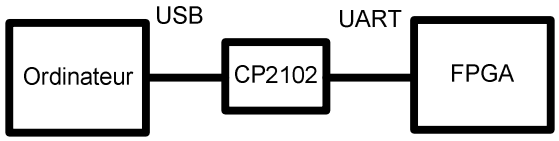

Figure 12. Circuit de communication UART

Dans le laboratoire, vous concevrez un module qui transmettra la lettre 'A' du FPGA à l'ordinateur lorsqu'un bouton est appuyé sur la plaquette. Quand ce bouton est pressé, l'information sera transmise et sera interprétée par le logiciel PUTTY. Si le format des données est bon, le logiciel devrait le comprendre et devrait afficher 'A' à l'écran.

Le protocole UART est un protocole série qui transmet l'information, bit par bit, sur un seul fil. Les données sont regroupées par ensemble de 7 ou 8 bits et sont envoyées par le lien série. Quand chaque groupe de 7 ou 8 bits est transmis, il est toujours précédé d'un START bit et il est suivi d'un STOP bit. L'ensemble de tous ces bits constitue une trame (« packet » en anglais).

Pour que la communication se fasse bien, il faut que les 2 côtés de la communication utilisent les mêmes paramètres. Le premier paramètre est la vitesse de communication. Il est possible de communiquer à des vitesses différentes. Pour les buts du laboratoire, choisissons une vitesse de 57600 bits par seconde. Les 3 autres caractéristiques que nous allons configurer sont le nombre de bit par trame, le bit de parité et la valeur du STOP bit. Pour ce laboratoire, nous allons utiliser 8 bits de données par trame, nous n'utiliserons aucun bit de parité et notre STOP bit sera égal à 1. De façon plus concise, on dit que c'est du « 8N1 », où 8 représente le nombre de bit, N veut dire « No parity » (aucun bit de parité) et 1 représente la valeur du STOP bit.

Dans le format 8N1, chaque trame a la forme suivante :

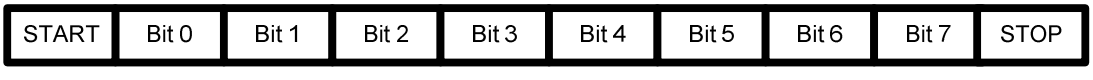

Figure 13. Trame de communication UART

Une trame en 8N1 est constituée de 10 bits, dont 8 contiennent de l'information. Il y aura toujours un START bit au début pour indiquer que c'est le début d'une trame. Ce START bit est toujours égal à 0. Il y aura toujours un STOP bit à la fin de la trame et nous avons convenu que sa valeur était égale à 1. Entre le START bit et le STOP bit on retrouve les données à envoyer. **Il est à noter que le bit suivant le START bit est le bit le MOINS significatif et que celui qui est envoyé avant le STOP est le bit le PLUS significatif**. Pour rendre les choses plus claires, utilisons un exemple. Quand on transmet des caractères (du clavier par exemple) en UART, ça se fait sous forme ASCII. Chaque caractère est représenté par un nombre de 8 bits et c'est ce nombre qui est envoyé. Si nous voulions envoyer le caractère '1' par UART (si nous pesions sur la touche '1' sur le clavier), il est possible de voir dans une TABLE ASCII (à trouver sur Google) que ce caractère correspond au chiffre $(31)_{16}$. Alors, si je voulais transmettre le caractère '1', il faudrait que je transmette $(31)_{16}$. La trame UART dans ce cas ressemblerait à ceci :

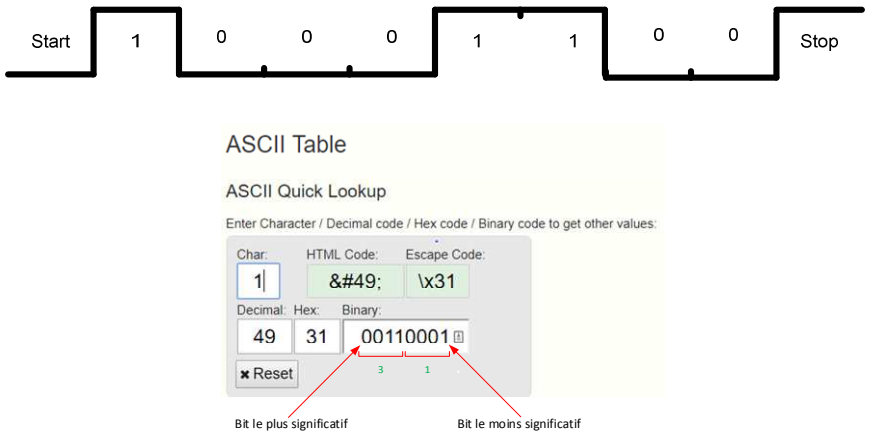

Figure 14. Séquence et table ASCII pour le nombre 1.

Pour une transmission à 57600bits par seconde, il faudrait que chaque bit dure 1/57600 seconde. Quand aucune information n'est transmise, la ligne devrait être à l'état 1. De façon plus explicite, la transmission du caractère '1' ressemblerait à ceci :

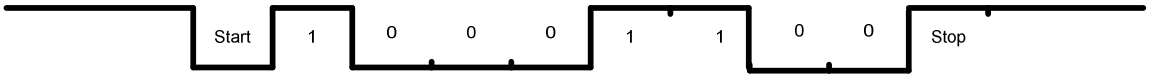

Figure 15. Diagramme de transmission du caractère « 1 »

La figure 15 est semblable à celle qui la précède sauf que nous montrons explicitement que, lorsqu'aucune donnée n'est envoyée, que la ligne devrait être à 1.

Pour connecter votre plaquette à l'ordinateur, vous passez par l'intermédiaire de la puce CP2102. Le laboratoire à l'UQAC possède plusieurs cartes qui utilisent ces puces et qui vous demandent un effort minime pour vous en servir. Une de ces cartes est montrée à la

figure suivante. Sur cette carte il y a 3 broches (RXD, TXD et GND) et un connecteur (USB) qui nous intéressent. Les connecteurs RXD et TXD représentent les broches pour la réception des données et la transmission des données respectivement. Le GND c'est la masse qu'il faut connecter aux masses des autres circuits pour que le système puisse bien fonctionner.

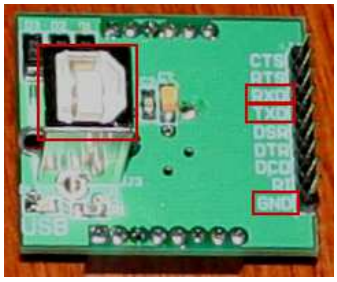

Figure 16. Circuit imprimé de la communication série (UART).

5. Travail demandé

Le but du laboratoire est de concevoir un module qui est capable d'envoyer le caractère 'A' par le port UART à un ordinateur. Il faut donc commencer par connaitre la séquence à envoyer lorsque nous voulons envoyer ce caractère. Nous savons que l'envoi du caractère 'A' se fait sous forme de code ASCII. La première étape est de faire une recherche sur internet pour trouver la table ASCII et identifier à quoi correspond le caractère 'A'.

1. Quelle est la valeur ASCII de la lettre 'A' en hexadécimale?
2. Déterminez la séquence de 10 bits à envoyer lorsque vous voulez envoyer la lettre 'A'.

Pour la prochaine étape, il s'agit de créer les blocs nécessaires pour faire la transmission du UART. Une des étapes importantes est la création d'une horloge qui nous permet d'envoyer les données à 57600 bits par seconde. Puisque nous n'avons pas réellement touché à la conception des compteurs, ce bloc vous sera fourni. Ce bloc utilise l'horloge de 50MHz (pin P11) pour donner une horloge de 57600Hz a la sortie.

Il existe plusieurs manières de faire le système de transmission. Une façon de le faire est d'utiliser les machines à états.

3. Quels sont les états nécessaires pour faire un système qui transmet 'A' en UART lorsqu'un bouton est à 1? Combien d'états y a-t-il?

Vous pouvez, par exemple, avoir un état initial où rien ne se passe. Évidemment, quand rien ne se passe en UART, il faut continuellement envoyer 1 à la sortie. Par la suite, si

l'usager appuie le bouton pour envoyer 'A', notre machine à états passe à l'état *envoie_start_bit* où la donnée transmise est 0 (le START bit). Par la suite, il va à l'état *envoie_bit0* où il envoie le bit le moins significatif de la lettre 'A'. Le processus continue jusqu'à ce qu'on se rende au STOP bit. Une fois le STOP bit envoyé, il retourne à l'état initial. (NOTE : vous pouvez peut-être remarquer qu'il n'est pas nécessaire d'avoir un état pour le STOP bit parce qu'il fait la même chose que l'état initial).

Une fois complété, allez chercher un module avec le CP2102 (UART), un câble USB et la plaquette de prototypage DE-10. Pour connecter le module CP2102 au FPGA, connectez les masses ensemble et connectez la sortie du FPGA à la broche RX du module. Ouvrez le programme PUTTY, et tentez d'établir la communication avec le FPGA. Pour ce faire, assurez-vous d'avoir le bon port de communication, la bonne vitesse et les bons paramètres (8N1). Demandez de l'aide si vous n'êtes pas certains.

Montrez au chargé de laboratoire que ça fonctionne.

NOTE : En connectant le module CP2102 a l'ordinateur, il se peut que le pilote de périphérique soit déjà présent. Dans ce cas, il est important d'aller voir sa configuration pour s'assurer que la vitesse de transmission soit de 57600 bits par seconde, 8 bits, pas de parité et 1 stop bit. En même temps, notez le nom du port auquel il est associé. Par exemple, il pourrait être associé au port COM4. C'est une information qui sera importante quand vous ouvrirez le logiciel PUTTY. Vous aurez à indiquer au logiciel que vous désirez utiliser le port COM4 (ou celui qui était indiqué quand vous avez examiné la configuration. Pour vérifier la configuration, allez dans Control Panel, Device Manager et sous la section Universal Serial Bus Controllers, trouvez celui qui correspond au CP2102 et faites un clic de droite et allez voir les propriétés.