ASR1 – TD4 : Prenons le bus

{ Andreea.Chis, Matthieu.Gallet, Bogdan.Pasca } @ens-lyon.fr

9 et 10 Octobre 2008

Le bus I^2C est un bus de communication sur seulement deux fils (plus la masse), très utilisé en électronique, par exemple sous les noms SMBus (dans les cartes mères) ou DCC (pour les communications entre moniteur et carte vidéo). Malgré le nombre de fils très restreint, le bus I^2C a tout d'un grand ; en particulier, il permet d'avoir plusieurs maîtres, et de gérer automatiquement les conflits sans utiliser d'arbitre externe. Nous allons ici essayer de retrouver le fonctionnement de ce bus.

Le bus I^2C se base sur une logique en non-OU câblé (on parle aussi de drain ouvert, en CMOS) : au repos une résistance ramène chacun des deux fils au niveau logique 1, et chaque système connecté au bus peut seulement forcer un fil au niveau 0. Chaque système peut lire l'état du bus en permanence.

1 Maîtres et zesclaves

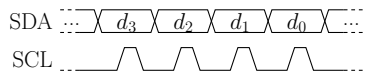
1. On suppose dans un premier temps qu'il y a un unique maître sur le bus, que chacun sait de qui il s'agit, et que lui seul à le droit d'écrire sur le bus (de mettre un fil à 0). Comment le maître peut-il envoyer une information (une suite de bits) à tout le monde ? Proposez un protocole simple qui utilise les deux fils.

Réponse :

Un des deux fils va servir à communiquer les données proprement dites, et l'autre va indiquer quand les données du premier fil seront valides, c'est-à-dire qu'elles pourront être lues par les esclaves.

*Les deux fils du protocole I^2C se nomment ainsi SDA (*Signal DAta*) et SCL (*Signal CLock*) : le fil SDA est le fil de données, ces données étant lues lorsque SCL passe à* 1*.*

On trouve ainsi le chronogramme suivant (notez que les bits sont transmis poids forts en tête) :



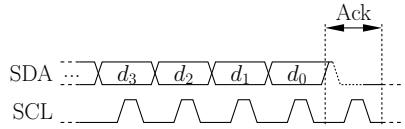
2. Les transmissions sont toujours faites octet par octet. Comment un esclave peut-il dire s'il a bien reçu l'octet ?

Réponse :

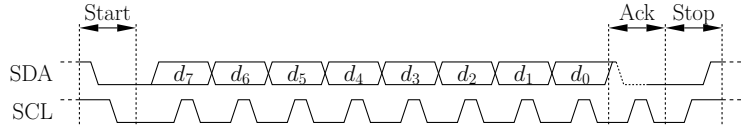
Il faut demander un acquittement Ack *à l'esclave. Ceci est fait en laissant le fil SDA revenir à* 1 *pendant une phase basse de l'horloge SCL. Pendant ce temps-là, l'esclave, lui, doit forcer SDA à* 0 *pour signaler l'acquittement. La sortie étant à drain ouvert, c'est bien un* 0 *qui est maintenu sur le fil SDA.*

Lorsque le maître relâche l'horloge SCL à 1*, il peut alors vérifier que la ligne SDA contient bien un* 0*, signifiant ainsi l'acquittement de l'esclave.*

D'où le chronogramme (le bit forcé par l'esclave est représenté en poitillés) :



3. Comment le maître peut-il signaler qu'il *prend* le bus et qu'il le *libère* ?
Réponse :
Pour l'instant, le fil SDA n'a le droit de changer d'état que sur un état à 0 *de SCL. Les codes* Start *et* Stop*, signalant respectivement prise et libération du bus, correspondent par contre à des transitions de SDA sur l'état haut de l'horloge.*
Ainsi, lorsque le bus est libre (les deux fils sont à 1*), le maître initie une communication en forçant un* 0 *sur SDA, alors que SCL est toujours à* 1*. Il force ensuite SCL à* 0 *et peut ainsi commencer à émettre des bits sur le bus.*
De même, pour libérer le bus, le maître relâche d'abord SCL qui remonte à 1*, puis enfin SDA.*

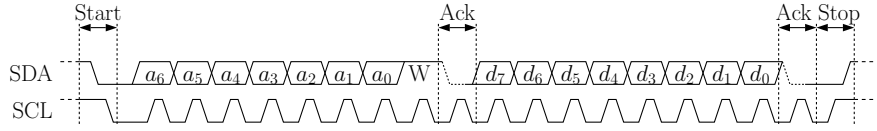


4. Il peut y avoir un grand nombre d'esclaves sur un même bus I^2C. Comment le maître peut-il préciser à qui il s'adresse ? Et comment peut-il préciser s'il veut envoyer ou recevoir une donnée ?
Réponse :
Il suffit pour le maître de transférer tout d'abord 7 *bits d'adresse puis un bit spécifiant si la transmission soit être un envoi (*1*) ou une réception (*0*).*

5. Que doit-on rajouter au bus pour pouvoir gérer plusieurs maîtres ?
Réponse :
Rien ! Le bus est prêt tel quel à supporter plusieurs maîtres. En fait, chaque composant est à la fois maître et esclave : il acquiert le statut de maître lorsqu'il initie une communication.

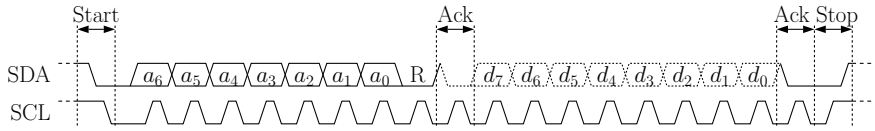
2 Chronogrammes

1. Dessinez des chonogrammes pour l'envoi et la demande d'un octet par le maître, en précisant bien qui du maître ou de l'esclave force quel niveau.
Réponse :
Dans le cas d'un envoi, le maître envoie d'abord une condition Start*, puis transmet d'abord l'adresse ($a_6 \dots a_0$) de l'esclave et un bit à* 1 *pour signaler un envoi sur le bus. Après un premier acquittement, il envoie l'octet en question ($d_7 \dots d_0$). Enfin, il clôt la communication après le second acquittement.*



Pour la réception d'un octet, on procède de la même manière, en envoyant l'adresse de l'esclave puis un bit à 0 *pour signaler une lecture sur le bus. Après le premier acquittement, la main est*

passée à l'esclave qui écrit la donnée sur SDA. Le maître garde cependant le contrôle sur l'horloge SCL. Enfin, le maître envoie son acquittement puis ferme la connexion.



2. Dessinez le chronogramme d'un maître qui lit 3 octets consécutifs d'une RAM (l'esclave).
Réponse :
Comme toujours, le maître écrit d'abord une condition Start*, puis l'adresse de l'esclave ainsi que le bit signalant un envoi. Après l'acquittement de l'esclave, il transmet alors l'adresse (sur un octet) à laquelle il souhaite lire dans la mémoire.*

Après le second acquittement, le maître envoie à nouveau une condition de départ (sans passer par Stop*) pour ne pas se faire prendre la main par un autre composant relié au bus. Il envoie alors à nouveau l'adresse de la mémoire, puis un bit à* 0 *signalant que le maître attend des données de la part de l'esclave.*

Enfin, l'esclave transmet les trois octets demandés, chaque octet étant suivi par un acquittement du maître.

3. Comment un esclave lent peut-il forcer le maître à ralentir ? Dessinez un chronogramme.
Réponse :
Il suffit de forcer l'horloge SCL à 0 *aussi longtemps que nécessaire. En effet, le maître décide de passer à la donnée suivante une fois l'horloge repassée à* 1*. Cependant, s'il ne détecte pas la remontée de l'horloge, il va attendre que l'esclave libère celle-ci avant de poursuivre.*

4. Comment détecter et gérer le conflit si plusieurs maîtres prennent le contrôle du bus en même temps ?
Réponse :
Si deux composants initient la connexion en même temps, ils ne sont pas forcément capables de détecter le conflit dès le début. Ils commencent donc à écrire des bits sur SDA. Tant que ces bits sont identiques, il est bien évidemment impossible pour eux de détecter quoi que ce soit non plus.

Par contre, dès que l'un des maîtres veut écrire un 1 *alors que l'autre veut envoyer un* 0*, le premier va relâcher le fil SDA pour qu'il remonte à* 1*, mais le second va forcer un* 0 *sur ce même fil. Le premier composant peut alors réaliser qu'il n'était pas le seul à émettre, et donc interrompre sa transmission.*

Au cas où les deux composants enverraient exactement le même message, le conflit ne sera pas détecté, mais d'un autre côté, puisque les messages étaient les mêmes, ce n'était pas véritablement un conflit.

Si vous voulez en savoir plus sur le bus I^2C, vous pouvez aller voir la page d'Aurélien Jarno qui nous a bien servis pendant la rédaction de ce TD : `http://www.aurel32.net/elec/i2c.php`.