Le bus I2C

1) Présentation :

1.1) Historique :

Le bus I2C (***Inter Integrated Circuit***) a été dévellopé au début des années 80 par Philips semiconductors pour permettre de relier facilement à un microprocesseur les différents circuits d'un téléviseur moderne.

1.2) Caractèristiques :

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grace à seulement trois fils : Un signal de donnée (SDA), un signal d'horloge (SCL), et un signal de référence électrique (Masse).

Ceci permet de réaliser des équipements ayants des fonctionnalités très puissantes (En apportant toute la puissance des systèmes microprogrammés) et conservant un circuit imprimé très simple, par rapport un schéma classique (8bits de données, 16 bits d'adresse + les bits de contrôle).

Les données sont transmises en série à 100Kbits/s en mode standard et jusqu'à 400Kbits/s en mode rapide. Ce qui ouvre la porte de cette technologie à toutes les applications où la vitesse n'est pas primordiales.

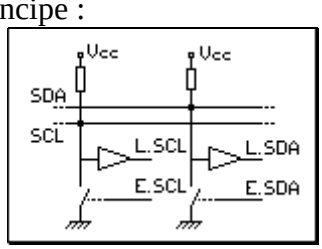
De nombreux fabricants ayant adopté le système, la variété des circuits disponibles diposant d'un port I2C est énorme : Ports d'E/S bidirectionnels, Convertisseurs A/N et N/A, Memoires (RAM, EPREM, EEPROM, etc...), Circuits Audio (Egaliseur, Contrôle de volume, ...) et autre drivers (LED , LCD , ...)

Le nombre de composants qu'il est ainsi possible de relier est essentiellement limité par la charge capacitive des lignes SDA et SCL : 400 pF .

1.3) Principe :

Afin de d'éviter les conflits électriques les Entrées/Sorties SDA et SCL sont de type "Collecteur Ouvert"

Voici un schéma de principe :



Structure d'E/S d'un module I2C.

1.4) D'autres bus trifilaires :

- CBus de Phillips est l'ancètre du bus I2C.
- SPI de Motorola.
- µWire de National SemiConductor.

Plusieurs circuits pouvant être branché en même temps sur le même bus, il a été nécessaire d'instaurer un protocole entre eux, afin d'éviter les problèmes dues à une prise de parole simultanée de différents modules. C'est le protocole I2C.

2) Le protocole I2C :

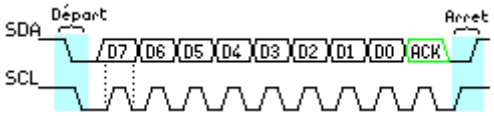
Le protocole I2C définit la succession des états logiques possibles sur SDA et SCL, et la façon dont doivent réagir les circuits en cas de conflits.

2.1) La prise controle du bus :

Pour prendre le controle du bus, il faut que celui-ci soit au repos (SDA et SCL à '1'). Pour transmettre des données sur le bus, il faut donc surveiller deux conditions particulières :

- La condition de départ. (SDA passe à '0' alors que SCL reste à '1')
- La condition d'arrèt. (SDA passe à '1' alors que SCL reste à '1')

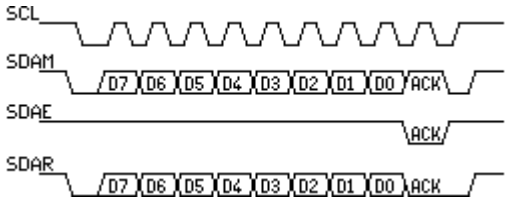
Lorsqu'un circuit, aprés avoir vérifié que le bus est libre, prend le controle de celui-ci, il en devient le **maître**. C'est lui qui génère le signal d'horloge.



Exemple de condition de départ et d'arrèt .

2.2) La transmission d'un octet :

Après avoir imposé la condition de départ, le maître applique sur SDA le bit de poid fort D7. Il valide ensuite la donnée en appliquant pendant un instant un niveau '1' sur la ligne SCL. Lorsque SCL revient à '0', il recommence l'opération jusqu'à ce que l'octet complet soit transmis. Il envoie alors un bit ACK à '1' tout en scrutant l'état réel de SDA. L'escave doit alors imposer un niveau '0' pour signaler au maître que la transmission s'est effectuée correctement. Les sorties de chacuns étant à collecteurs ouverts, le maître voie le '0' et peut alors passer à la suite.



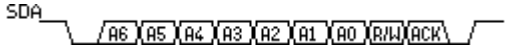
Exemple de transmission réussie.

Dans cet exemple :

- SCL : Horloge imposée par le maître.
- SDAM : Niveaux de SDA imposés par le maître.
- SDAE : Niveaux de SDA imposés par l'esclave.
- SDAR : Niveaux de SDA réels résultants.
-

2.3) La transmission d'une adresse :

Le nombre de composants qu'il est possible de connecter sur un bus I2C étant largement supérieur à deux, il est nécessaire de définir pour chacun une adresse unique. L'adresse d'un circuit, codée sur sept bits, est défini d'une part par son type et d'autre part par l'état appliqué à un certain nombre de ces broches (Voir §9). Cette adresse est transmise sous la forme d'un octet au format particulier.



Exemple d'octet d'adresse.

On remarque ici que les bits D7 à D1 représentent les adresse A6 à A0, et que le bit D0 et remplacé par le bit de R/W qui permet au maître de signaler s'il veut lire ou

écrire une donnée. Le bit d'acquitement ACK fonctionne comme pour une donnée, ceci permet au maître de vérifier si l'esclave est disponible.

Note 1: Cas perticulier des mémoires :

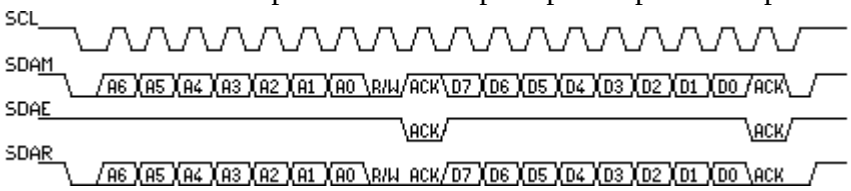
L'espace adressable d'un circuit de mémoire étant sensiblement plus grand que la plupart des autres types de circuits, l'adresse d'une information y est codée sur deux octets ou plus. Le premier représente toujours l'adresse du circuit, et les suivants l'adresse interne de la mémoire.

Note 2: Les adresses réservées.

Les adresses 00000XXX et 111111XX sont réservés à des modes de fonctionnement particuliers.

2.4) Ecriture d'une donnée :

L'écriture d'une donnée par le maître ne pose pas de problème particulier :



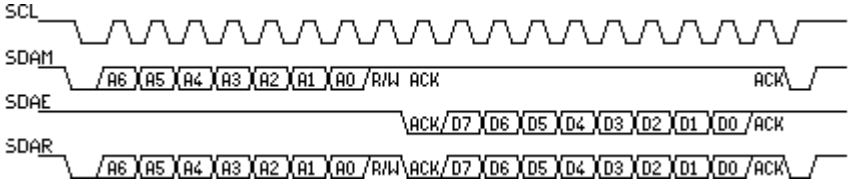
Exemple d'écriture d'une donnée.

Note : Cas perticulier d'utilisation d'ACK :

L'écriture d'un octet dans certains composants (Mémoires, microcontrôleur, ...) peut prendre un certain temps. Il est donc possible que le maître soit obligé d'attendre l'acquitement ACK avant de passer à la suite.

2.5) Lecture d'une donnée :

La lecture d'une donnée par le maître se caractèrise par l'utilisation spéciale qui faite du bit ACK. Après la lecture d'un octet, le maître positionne ACK à '0' s'il veut lire la donnée suivante (cas d'une mémoire par exemple) ou à '1' la cas échéant. Il envoie alors la condition d'arret.



Exemple de lecture d'une donnée.

3) La gestion des conflits :

3.1) Mise en situation :

La structure même du bus I2C a été conçu pour pouvoir y accueillir plusieurs maîtres. Se pose alors le problème commun à tout les réseaux utilisant un canal de communication unique : la prise de parole. En effet, chaque maître pouvant prendre possession du bus dès que celui-ci est libre, il existe la possibilité de que deux maîtres prennent la parole en même temps. Si cela ne pose pas de problème sur le plan électrique grace l'utilisation de collecteurs ouverts, il faut pouvoir détecter cet état de fait pour éviter la corruption des données transmises.

3.2) Principe :

Comme nous l'avons vu précedement, pour prendre le contrôle du bus, un maître potentiel doit d'abord vérifier que celui-ci soit libre, et qu'une condition d'arrèt ait bien été

envoyée depuis au moins 4,7µs. Mais il reste la possibilité que plusieurs maîtres prennent le contrôle du bus simultanement.

Chaque circuit vérifie en permanence l'état des lignes SDA et SCL, y compris lorsqu'ils sont eux même en train d'envoyer des données. On distingue alors plusieurs cas :

(1) Les différents maîtres envoyent les mêmes données au même moment :

Les données ne sont pas corrompues, la transmission s'effectue normalement, comme si un seul maître avait parlé. Ce cas est rare.

(2) Un maître impose un '0' sur le bus :

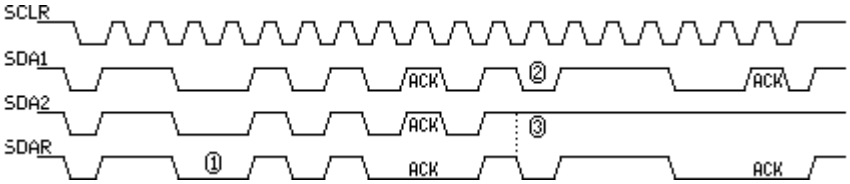
Il relira forcément '0' et continuera à transmettre. Il ne peut pas alors détecter un eventuel conflit.

(3) Un maitre cherche à appliquer un '1' sur le bus :

Si il ne relit pas un niveau '1', c'est qu'un autre maître a pris la parole en même temps. Le premier perd alors immédiatement le controle du bus, pour ne pas perturber la transmission du second. Il continue néanmoins à lire les données au cas celles-ci lui auraient été destinées.

3.3) Exemple :

Soit le chronogramme suivant :



Dans cet exemple :

- SCLR : Horloge résultante.
- SDA1 : Niveaux de SDA imposés par le maître n°1.
- SDA2 : Niveaux de SDA imposés par le maître n°2.
- SDAR : Niveaux de SDA réels résultants lus par les deux maîtres.

Analyse :

Le premier octet est transmis normalement car les deux maîtres imposent les même données. (Cas n°1). Le bit ACK est mis à '0' par l'esclave.

Lors du deuxième octet, le maître n°2 cherche à imposer un '1' (SDA2) , mais relit un '0' (SDAR), il perd alors le contrôle du bus et devient esclave (Cas n°3) . Il reprendra le contrôle du bus, lorsque celui-ci sera de nouveau libre.

Le maître n°1 ne voit pas le conflit et continue à transmettre normalement. (Cas n°2)

Au total, l'esclave à reçu les données du maître n°1 sans erreurs et le conflit est passé inaperçu.

4) Les nouvelles caractéristiques :

Afin de compenser quelques lacunes des pemières spécifications du bus I2C (qui datent de 1982), quelques nouvelles améliorations ont été apportées à partir de 1993 :

Le mode rapide : Le bus à désormais la capacité de transmettre des données jusqu'à une vitesse de 400 Kbit/s.

Des entrées à triggers de Shmitt : Afin de limiter la sensibilité au bruit.

La mise en haute impédance d'un cicuit non alimenté : Ceci évite de bloquer le bus si un périphérique n'est pas alimenté.

Extention à 10 bits de l'adressage des circuits :

L'adressage d'un circuits se fait maintenant sur 10 bits répartis dans deux octets d'adresse de la façon suivante :

5) Les adresses réservées :

Les addresses 0000 0xxx ne sont pas utilisées pour l'adressage de composants. Ils ont été résevés par Phillips pour effectuer certaines fonctions spéciales.

5.1) Adresse d'appel general :

Adresse : 0000 0000 .

Après l'émission d'un appel général, les circuits ayant la capacité de traiter ce genre d'appel émettent un acquitement.

Le deuxième octet permet definir le contenu de l'appel :

0000 0110 : RESET. Remet tout les registres de circuits connectés dans leur état initial (Mise sous tension). Les circuits qui le permettent rechargent leur adresse d'esclave.

0000 0010 : Les circuits qui le permettent rechargent leur adresse d'esclave.

0000 0100 : Les circuits définissant leur adresse de façon matériel reinitialisent leur adresse d'esclave.

0000 0000 : Interdit

xxxx xxx1 : Cette commande joue le rôle d'interruption. xxxx xxx peut être l'adresse du circuit qui a généré l'interruption.

5.2) Octet de Start :

Adresse : 0000 0001 .

Cet octet est utilisé pour synchroniser les périphériques lents avec les périphériques rapides.

5.3) Debut d'adressage CBus :

Adresse : 0000 001x .

L'emission de cet octet permet de rendre sourd tout les circuits I2C présent sur le bus. A partir de ce momont, on peut transmettre ce que l'on desire sur le bus, en utilisant par exemple un autre protocole. Le bus repasse en mode normal lors de la réception d'une condition d'arrèt.

5.4) Autre :

Adresses : 0000 0110 à 0000 1111 .

Ces octets ne sont pas définits et sont ignoré par les circuits I2C. Il peuvent être utilisé pour débugger un reseau multimaster.

D’après les pages sur le bus I2C trouvées sur le site :

http://www.planete.net/~surbanov/i2c/intro.html