Bus I²C

Table des matières

I2C (Inter-Integrated Circuit) est un bus informatique qui a émergé de la « guerre des standards » lancée par les acteurs du monde électronique. Conçu pour les applications de domotique et d'électronique domestique, il permet de relier facilement un microprocesseur et différents circuits.

1. Introduction

Dans les systèmes grands publics (téléviseurs, appareils hifi, lecteur CD...), les échanges de données entre les composants se faisait sur des bus parallèles.

Du fait de la sophistication des appareils toujours plus importante, ces liaisons parallèles devenaient de plus en plus nombreuses.

Dans le but de minimiser ces liaisons, par conséquent pour augmenter la fiabilité, a été crée le bus série I²C au début des années 80 par Philips.

D'autres fabricants ont adoptés le protocole I²C et de nombreux composants sont apparus (mémoires, convertisseurs, capteurs , micro - contrôleurs ...)

Le bus I²C fait partie des réseaux LAN.

Les normes I2C (Inter-Integrated Circuit) et SPI (Serial Peripheral Interface) ont été créées pour fournir un moyen simple de transférer des informations numériques entre des capteurs et des microcontrôleurs.

Les bibliothèques Arduino pour I2C et SPI facilitent l'utilisation de ces deux protocoles.

Le choix entre I2C et SPI est en général déterminé par les périphériques que l'on souhaite connecter. Certains appareils offrent les deux standards, mais habituellement un périphérique ou une puce ne supporte qu'une seule des deux normes.

I2C à l'avantage de n'avoir besoin que de deux connexions de signalisation (l'emploi de plusieurs périphériques sur les deux connexions est assez simple et on reçoit une confirmation que les signaux ont été correctement reçus).

L'inconvénient est que la vitesse des données est inférieure à celle de SPI et qu'elles ne peuvent voyager que dans un seul sens à la fois (Simplex), ce qui abaisse encore plus le débit si des communications bidirectionnelles sont nécessaires (Half Duplex).

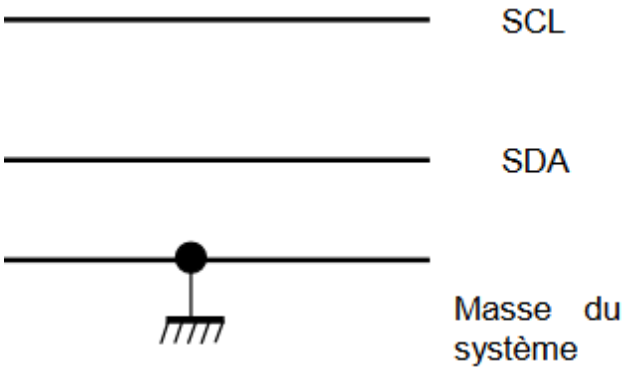
L'avantage de SPI est qu'il a un meilleur débit et qu'il a des connexions d'entrée et de sorties séparées, si bien qu'il peut envoyer et recevoir en même temps (Full Duplex). Il utilise une ligne supplémentaire par appareil pour sélectionner le périphérique actif et il faut donc plus de connexions si on a de nombreux appareils à connecter.

2. Propriétés physiques du bus I²C

2.1. Le support physique

Le bus I²C est de faible longueur. Il est réalisé par des pistes de cuivre sur circuit imprimé ou bien de manière filaire.

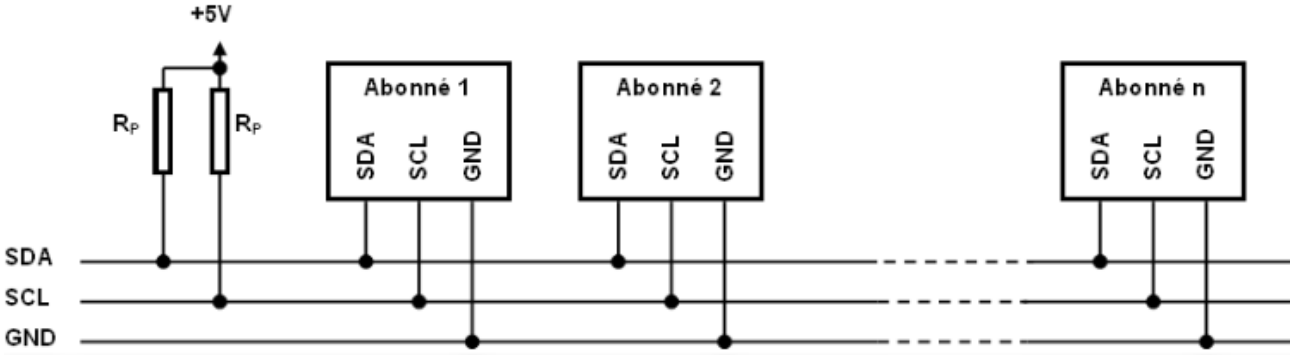
La transmission se fait sur 3 fils :



- SCL (Serial Clock) : horloge de cadencement de la communication.
- SDA (Serial Data) : signal de données
- Les périphériques I2C ont besoin d'une masse commune pour communiquer.

2.2. Synoptique d'une liaison I²C

La charge capacitive maximum des lignes limite le nombre d'abonnés pouvant être connectés au bus I²C. Cela limite aussi la longueur de la liaison (piste et/ou câble).



2.3. Arbitrage

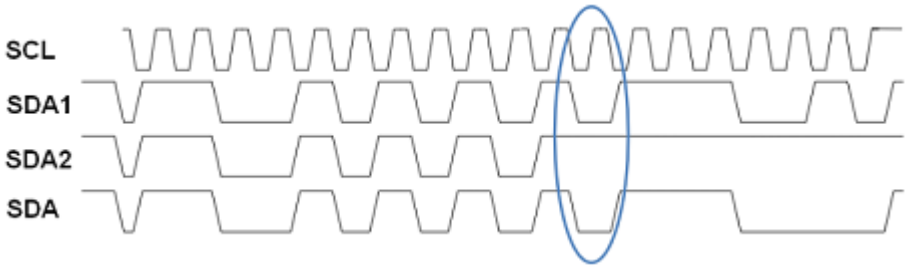
Le bus I²C est multi - maîtres. Dès que le bus est libre, plusieurs abonnés peuvent prendre la parole en même temps.

Pour éviter la transmission de données erronées, le maître relit la donnée qu'il a placée sur le bus.

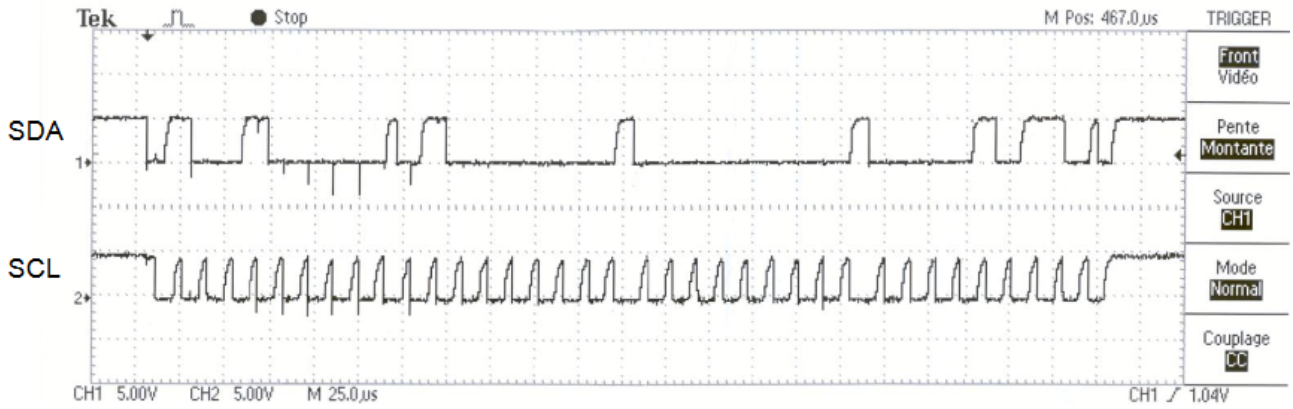
Dès que la donnée lue est différente de la donnée écrite, un autre maître a pris la parole en même temps. Il perd l'arbitrage et arrête d'émettre.

Exemple : Deux maîtres prennent la parole en même temps. Ils émettent les données SDA1 et SDA2. Tant qu'ils envoient la même donnée, rien ne se passe. On la retrouve sur la ligne SDA.

Lorsque les maîtres envoient une donnée différente, celui qui a placé un NL 1, lit sur la ligne SDA un NL 0. Il arrête d'émettre (SDA2 au NL 1) et l'autre maître continue la transmission.

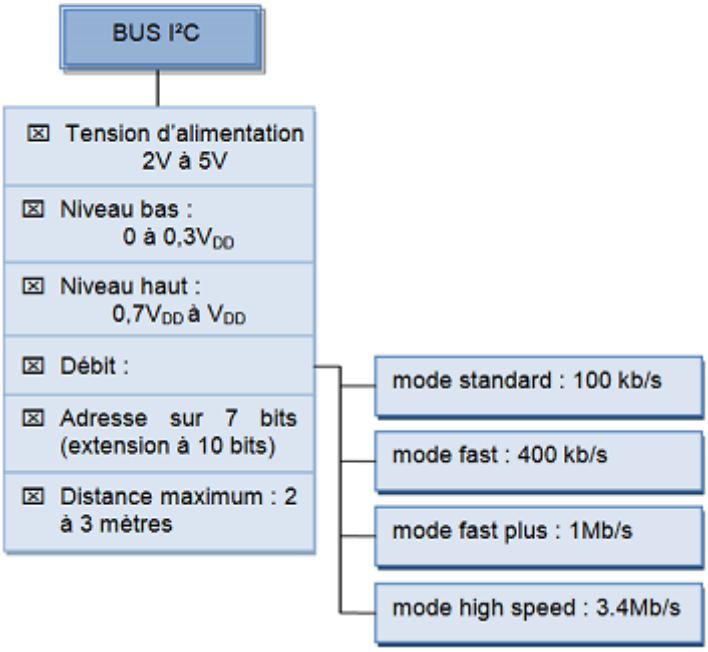


Le maître perdant l'arbitrage devient aussitôt récepteur, pour détecter si l'adresse émise par le maître gagnant est la sienne.



Oscillogrammes des signaux électriques mode standard 100 kbits/s

2.4. Version du bus



3. Le bus i2c

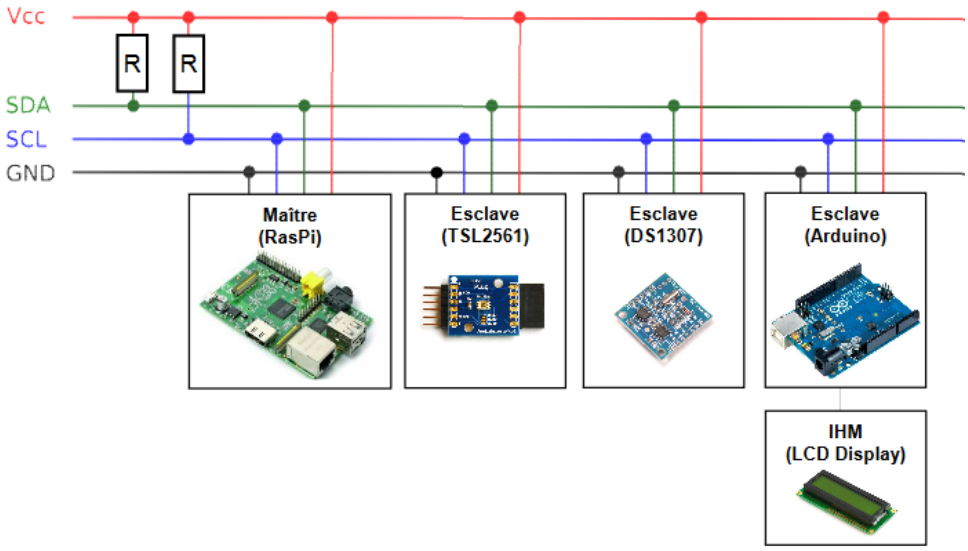
Les deux connexions du bus I2C se nomment SCL (Serial Clock Line) et SDA (Serial Data line).

Elles sont disponibles sur une carte standard Arduino en employant la broche analogique 5 pour SCL qui fournit un signal d'horloge, et la broche analogique 4 pour SDA, qui s'occupe du transfert des données (sur la Mega, il faut utiliser la broche 20 pour SDA et la broche 21 pour SCL). La carte Uno rev 3 a des broches supplémentaires qui dupliquent les broches 4 et 5.

Pour le Raspberry Pi type B, il s'agit respectivement des broches GPIO2 et GPIO3 pour SDA et SCL repérées sur le « Pi Cobbler ».

Un périphérique sur le bus I2C est considéré comme le périphérique maître. Son travail consiste à coordonner le transfert des informations entre les autres périphériques (esclaves) qui sont connectés.

Il ne doit y avoir qu'un seul maître qui contrôle les autres composants auxquels il est connecté. La figure ci-dessous montre un maître I2C avec plusieurs esclaves I2C.



Maître I2C avec plusieurs esclaves I2C

Les périphériques esclaves sont identifiés par leur numéro d'adresse. Chaque esclave doit avoir une adresse unique. Certains appareils I2C ont une adresse fixe alors que d'autres permettent que l'on configure leur adresse en définissant les broches à HIGH ou LOW ou en envoyant des commandes d'initialisation.

Remarques :

1. L'Arduino utilise des valeurs sur 7 bits pour spécifier les adresses I2C. Certaines notices techniques de périphériques emploient des adresses sur 8 bits, dans ce cas il faut diviser par deux pour obtenir la valeur correcte sur 7 bits.
2. La bibliothèque Arduino Wire masque toutes les fonctionnalités de bas niveau du I2C et permet l'emploi de commandes simples pour initialiser les appareils et communiquer avec eux.

Le bus I2C permet d'échanger des informations sous forme série avec un débit pouvant atteindre 100 kilobits/s ou 400 kilobits/s pour les versions les plus récentes.

Ses points forts sont les suivants :

- c'est un bus série bifilaire utilisant une ligne de donnée SDA et une ligne d'horloge SCL
- le bus est multi maîtres
- chaque abonné dispose d'une adresse codée sur 7 bits, on peut donc connecter simultanément 128 abonnés d'adresses différentes sur le même bus (sous réserve de ne pas le surcharger électriquement = la charge)
- un acquittement est généré pour chaque octet de donnée transféré
- un procédé permet de ralentir l'équipement le plus rapide pour s'adapter à la vitesse de l'équipement le plus lent lors d'un transfert ;
- le nombre maximal d'abonné n'est limité que par la charge capacitive maximale du bus qui peut être de 400 pF
- les niveaux électriques permettent l'utilisation des circuits en technologies CMOS, NMOS ou TTL

3.1. Protocole

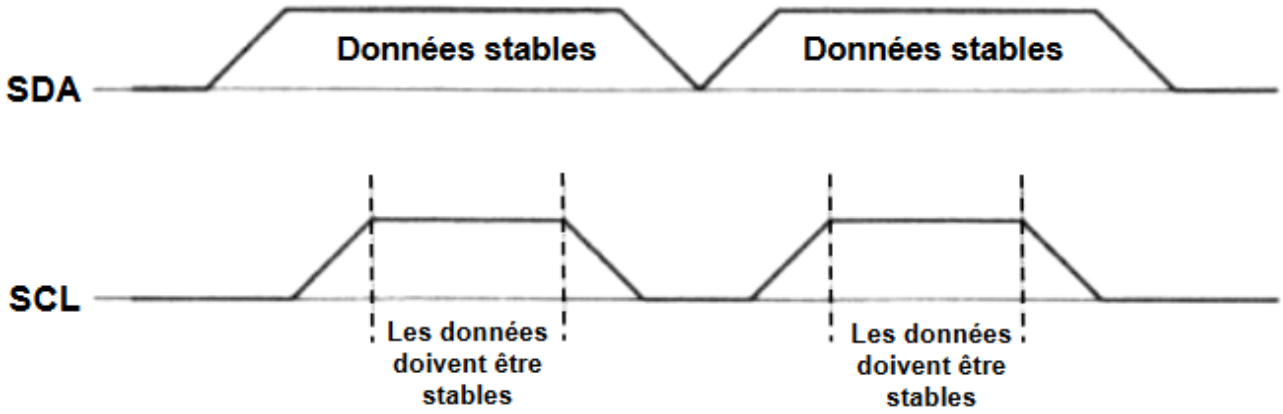
La communication sur le bus I²C ne peut se faire qu'entre 2 abonnés.

Lorsqu'un abonné prend le contrôle du bus, il devient le maître de la communication. Il génère le signal d'horloge SCL et communique avec un esclave. Selon le sens de la communication, il sera l'émetteur ou le récepteur.

Lorsque aucun abonné n'émet sur le bus, les lignes SDA et SCL sont au niveau haut qui est leur état de repos.

Quand une ligne (SDA ou SCL) est au repos (niveau 1), on peut la forcer à 0.

Quand une ligne (SDA ou SCL) est au niveau 0, on ne peut pas la forcer à 1.



Le chronogramme ci-dessus résume le principe fondamental d'un transfert de données :

Une donnée n'est considérée comme valide sur le bus que lorsque le signal SCL est à l'état haut.

L'émetteur doit donc positionner la donnée à émettre lorsque SCL est à l'état bas et la maintenir tant que SCL est à l'état haut.

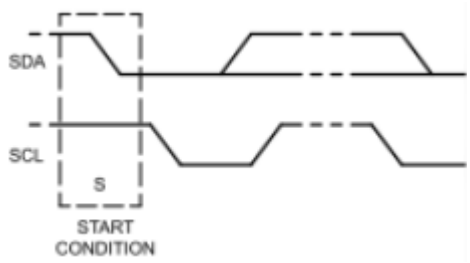
Comme la transmission s'effectue sous forme série, une information de début et de fin doit être prévue. L'information de début se nomme START et l'information de fin STOP.

3.1.1. Début de communication

Un abonné prend le contrôle du bus I²C en émettant une condition de départ :

1. Niveau haut sur SCL
2. Front descendant sur SDA

Cet abonné devient le maître.

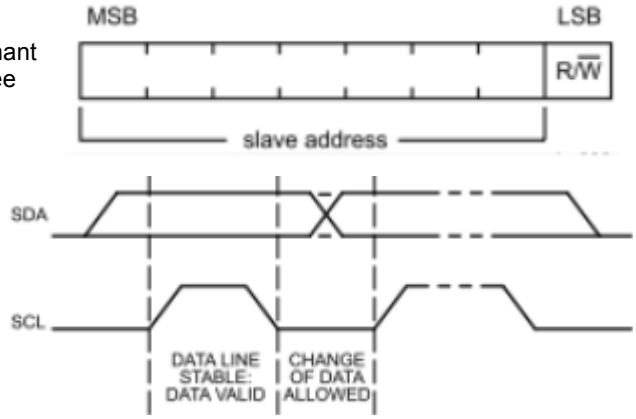


3.1.2. Transmission de l'adresse

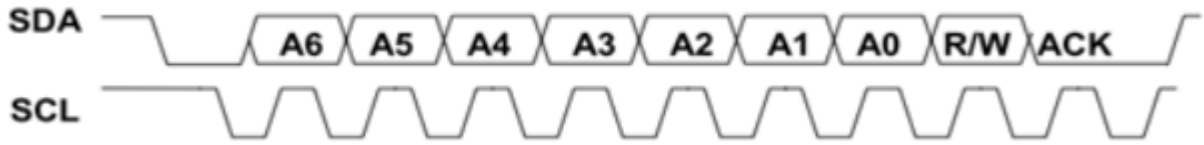
Après avoir pris le contrôle, le maître transmet un octet contenant l'adresse de l'esclave (sur 7 bits) ainsi que l'opération effectuée (écriture ou lecture).

Lecture (NL 1), Écriture (NL 0)

La transmission d'un bit se fait lorsque le signal SCL est au niveau haut :



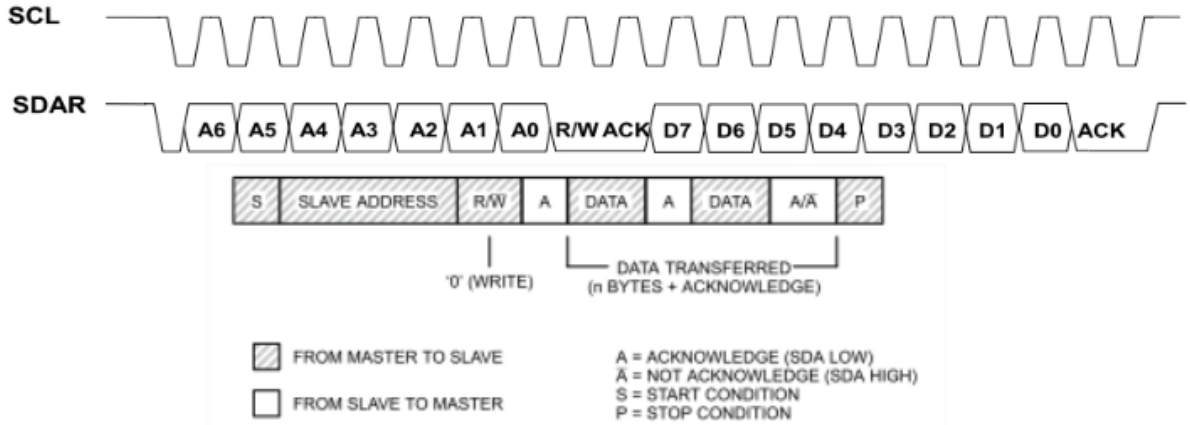
Lorsque l'esclave a détecté son adresse, il émet un bit d'acquittement (ACK) au niveau logique bas.



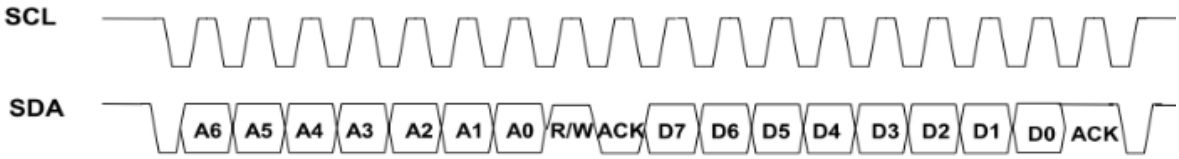
3.1.3. Transmission des données

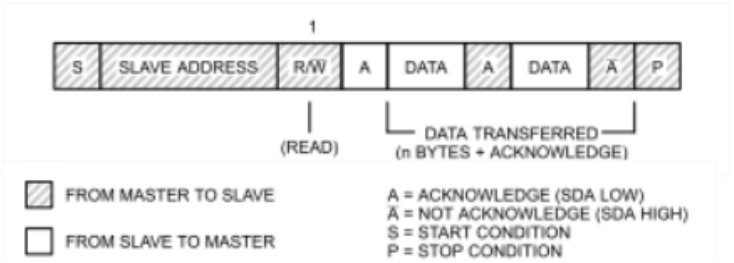
Deux cas se présentent :

1. Le maître envoie des données à l'esclave . A la fin de la transmission de chaque octet, l'esclave émet un acquittement.

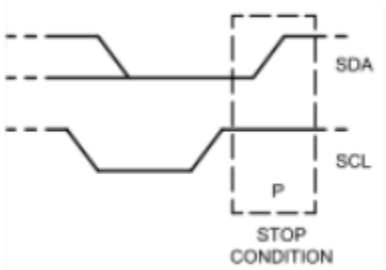


2. L'esclave envoie des données au maître . A la fin de la transmission d'un octet, le maître émet un acquittement s'il veut recevoir encore un octet ou bien un non acquittement (NL 1) s'il a terminé de recevoir.





3.1.4. Fin de communication



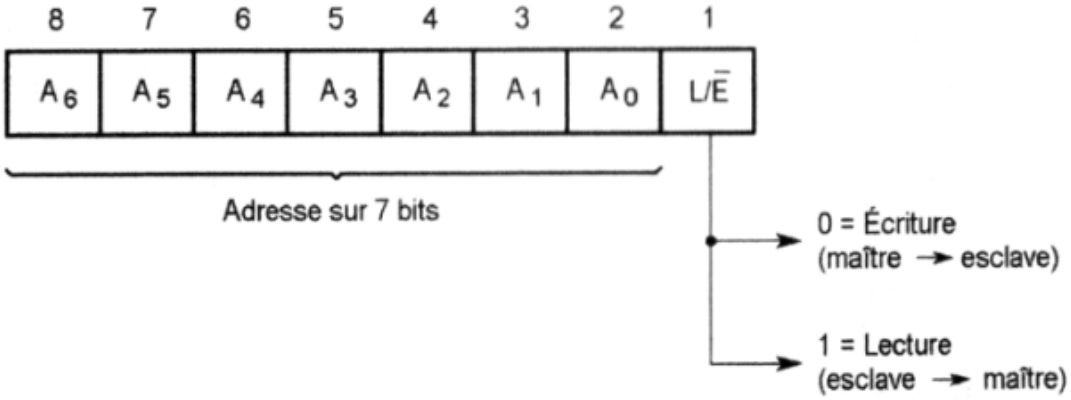
Pour terminer la communication, le maître émet une condition d'arrêt.

1. Niveau haut sur SCL
2. Front montant sur SDA

Tous les abonnés sont alors déconnectés du bus. SDA et SCL sont au niveau haut.

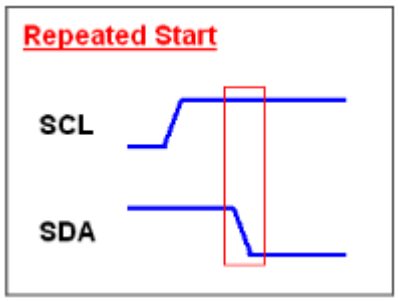
3.2. En-tête

La figure page suivante montre le contenu du premier octet qui est toujours présent en début d'échange. Ses sept bits de poids forts contiennent l'adresse du destinataire, ce qui autorise 128 combinaisons différentes. Le bit de poids faible indique si le maître va réaliser une lecture ou une écriture. Si ce bit est à zéro, le maître va écrire dans l'esclave ou lui envoyer des données. Si ce bit est à un, le maître va recevoir des données de l'esclave.



Contenu de l'octet d'en-tête de l'échange sur le bus I2C

Lorsqu'un maître désire effectuer plusieurs échanges à destination d'esclaves d'adresses différentes, il n'est pas obligé de terminer le premier échange par une condition d'arrêt mais peut les enchaîner en générant une condition de départ dès la fin d'un échange.



3.3. Exemple avec l'esclave DS1307

Le circuit Dallas DS1307 est une horloge temps réel (Real Time Clock), qui fournit secondes, minutes, heures, jours, dates, mois et années. Les années bissextiles sont prises en compte (jusqu'en 2100).

Le DS1307 travaille dans le mode standard (fréquence d'horloge de 100 kHz).

L'adresse I2C (7 bits) du DS1307 est : 1101000 (adresse fixée par le constructeur et non modifiable).

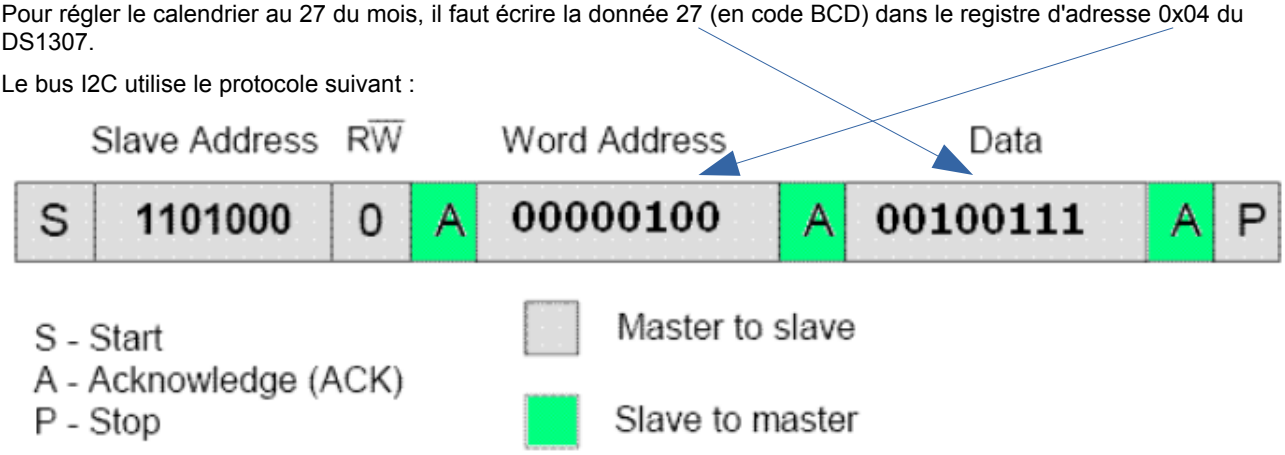
3.3.1. Exemple d'écriture du DS1307

L'émetteur est le maître et le récepteur est l'esclave.

Le registre d'adresse 0x04 du DS1307 contient la date (voir datasheet du DS1307).

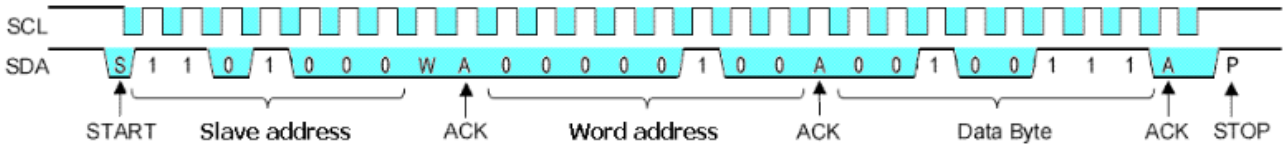
Pour régler le calendrier au 27 du mois, il faut écrire la donnée 27 (en code BCD) dans le registre d'adresse 0x04 du DS1307.

Le bus I2C utilise le protocole suivant :



1. Pour initier le dialogue, le maître crée une condition Start.
2. Le maître envoie l'adresse de l'esclave (1101000) suivi du bit 0 (bit Write).
3. L'esclave répond (accusé de réception : bit ACKnowledge).
4. Le maître envoie l'adresse du registre (04h) à écrire.
5. L'esclave répond (accusé de réception : bit ACKnowledge).
6. Le maître envoie la donnée (27) à écrire.
7. L'esclave écrit la donnée puis envoie un accusé de réception (bit ACKnowledge).
8. Le maître termine le dialogue avec une condition Stop.

Le bus I2C est maintenant libre (SCL = 1, SDA = 1 = niveaux de repos).



Chronogramme complet de l'échange sur bus I2C avec l'esclave DS1307

D'après l'exemple ci-dessus, combien de temps faut-il pour le transfert en écriture ?

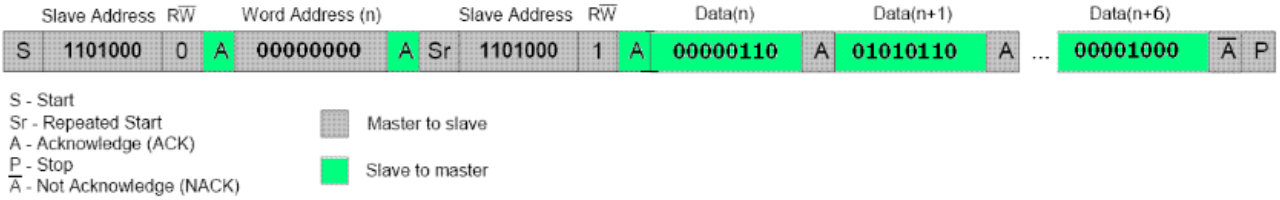
.....

3.3.2. Exemple de lecture du DS1307

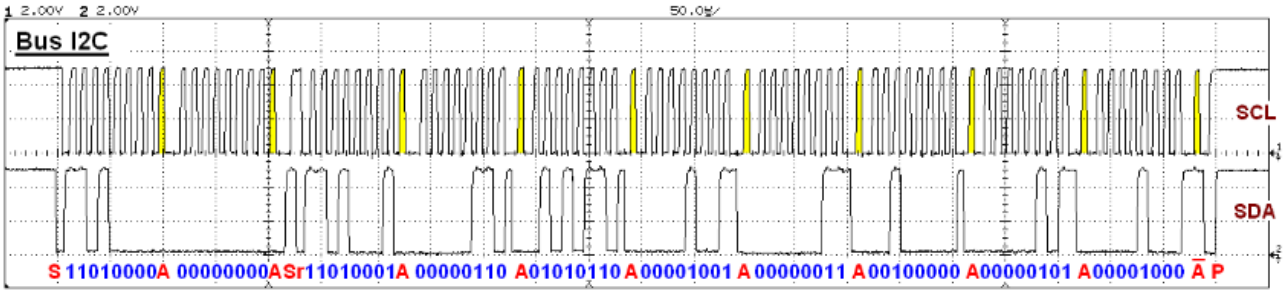
L'émetteur est l'esclave et le récepteur est le maître.

Les registres d'adresses 00h à 06h du DS1307 contiennent respectivement les secondes, minutes, heures, jours, dates, mois et années (voir datasheet du DS1307).

Voici comment lire, d'une seule traite, le contenu des registres d'adresses 00h à 06h du DS1307 :



Chronogramme correspondant :



Chronogramme complet de l'échange sur bus I2C avec l'esclave DS1307

1. Pour initier le dialogue, le maître crée une condition Start.
2. Le maître envoie l'adresse de l'esclave (1101000) suivi du bit 1 (bit Write).
3. L'esclave répond (accusé de réception : bit ACKnowledge).
4. Le maître envoie l'adresse du registre (0x00) à lire.
5. L'esclave répond (accusé de réception : bit ACKnowledge).
6. Le maître émet une condition Repeated Start.
7. Le maître envoie l'adresse de l'esclave (1101000) suivi du bit 1 (bit Read).
8. L'esclave répond (accusé de réception : bit ACKnowledge).
9. L'esclave envoie le contenu du registre d'adresse 0x00 au maître.
10. Le maître répond (accusé de réception : bit ACKnowledge).
11. L'esclave envoie le contenu du registre d'adresse 0x01 (automatiquement incrémenté) au maître.
12. Le maître répond (accusé de réception : bit ACKnowledge).
13. L'esclave envoie le contenu du registre d'adresse 0x02 (automatiquement incrémenté) au maître.
14. Le maître répond (accusé de réception : bit ACKnowledge).

    .....

    .....

21. L'esclave envoie le contenu du registre d'adresse 0x06 (automatiquement incrémenté) au maître.
22. Le maître répond (accusé de réception : bit Not ACKnowledge).
23. Le maître termine le dialogue avec une condition Stop.

Le bus I2C est maintenant libre (SCL = 1, SDA = 1 = niveaux de repos).

Retrouver précisément la date et l'heure à l'aide du chronogramme complet de l'échange sur bus I2C avec l'esclave DS1307.

.....

4. Compléments

Référence bibliothèque Arduino Wire : http://www.arduino.cc/en/Reference/Wire

Protocole et spécifications du bus I2C : http://electro8051.free.fr/I2C/busi2c.htm