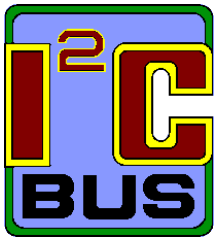
Bus I2C

Table des matières

I2C est un bus de données qui a émergé de la « guerre des standards » lancée par les acteurs du monde électronique. Conçu par Philips pour les applications de domotique et d'électronique domestique, il permet de relier facilement un microprocesseur et différents circuits, notamment ceux d'une télévision moderne : récepteur de la télécommande, réglages des amplificateurs basses fréquences, tuner, horloge, gestion de la prise péritel, etc.



1. Introduction

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). Il faut aussi connecter des résistances « pull-up » aux connexions pour s'assurer de la fiabilité de la transmission des signaux.

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.

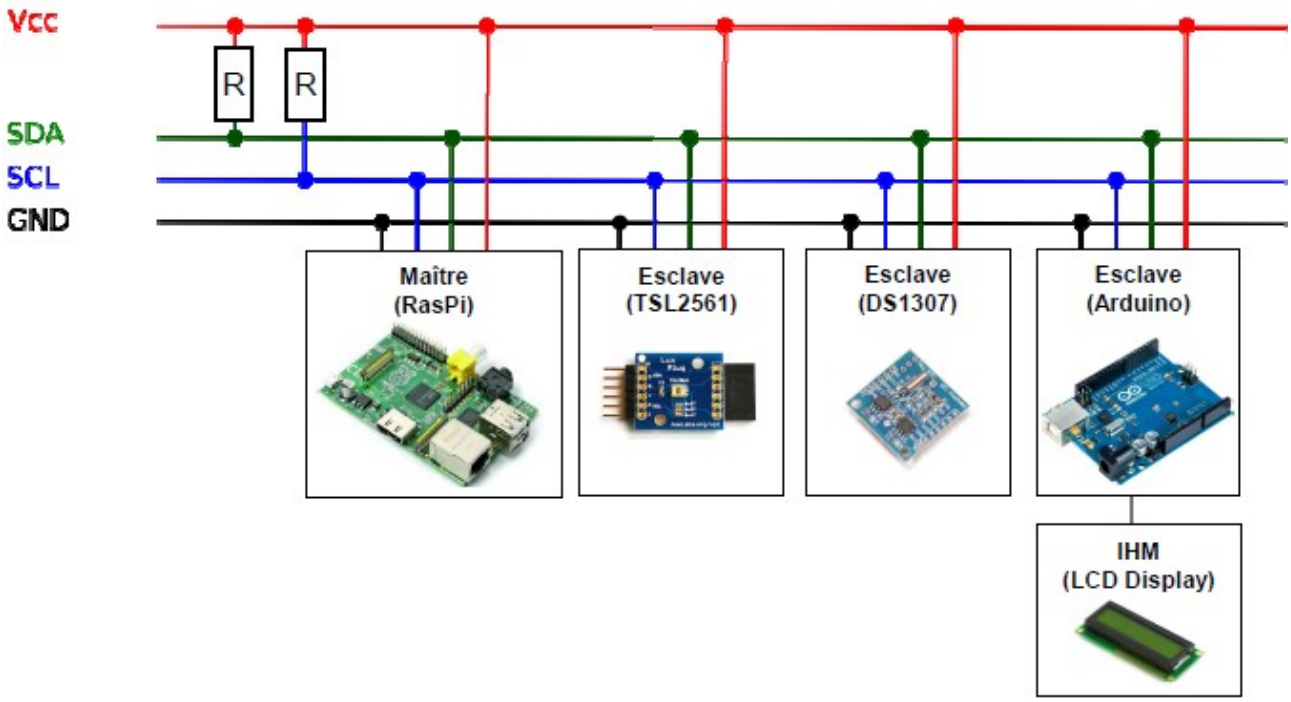
2. Le bus I2C

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 I2C ont besoin d'une masse commune pour communiquer.

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 :

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.

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. Protocole

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.

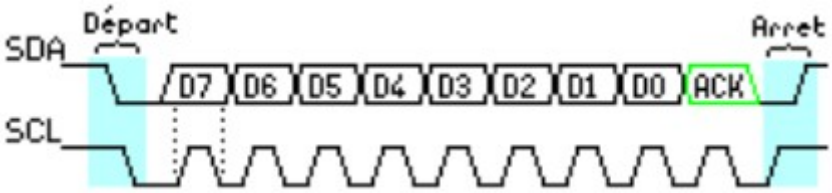
3.1. La prise contrôle du bus

Pour prendre le contrôle 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'arrêt. (SDA passe à '1' alors que SCL reste à '1')

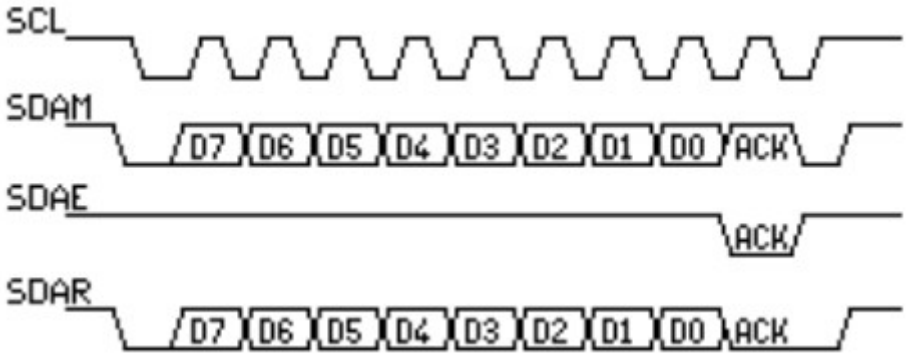
Lorsqu'un circuit, après avoir vérifié que le bus est libre, prend le contrôle de celui-ci, il en devient le maître . C'est lui qui génère le signal d'horloge.



Exemple de condition de départ et d'arrêt.

3.2. La transmission d'un octet

Après avoir imposé la condition de départ, le maître applique sur SDA le bit de poids 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'esclave doit alors imposer un niveau '0' pour signaler au maître que la transmission s'est effectuée correctement. Les sorties de chacun étant à collecteurs ouverts, le maître voie le '0' et peut alors passer à la suite.



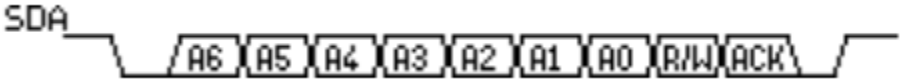
Exemple de transmission réussie.

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.

3.3. 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. 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’acquittement ACK fonctionne comme pour une donnée, ceci permet au maître de vérifier si l'esclave est disponible.

Remarques :

1. Cas particulier 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.

2. Les adresses réservées :

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

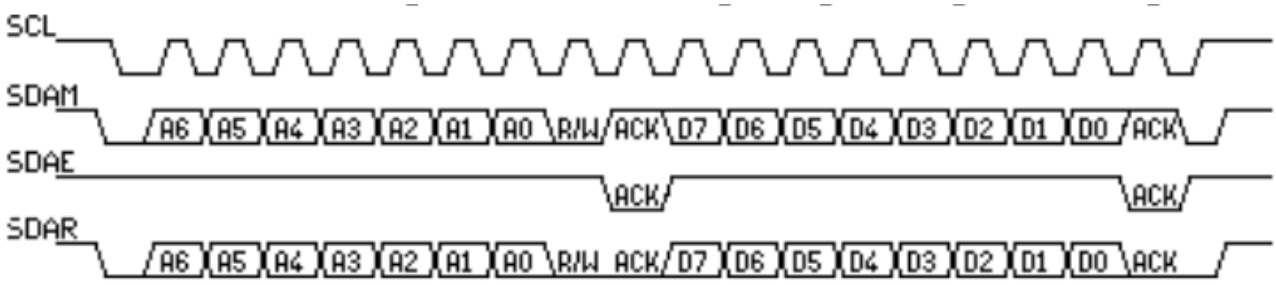
   Après l'émission d'un appel général, les circuits ayant la capacité de traiter ce genre d'appel émettent un acquittement.

   Le deuxième octet permet définir 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 réinitialisent 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.

3.4. Écriture d'une donnée

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

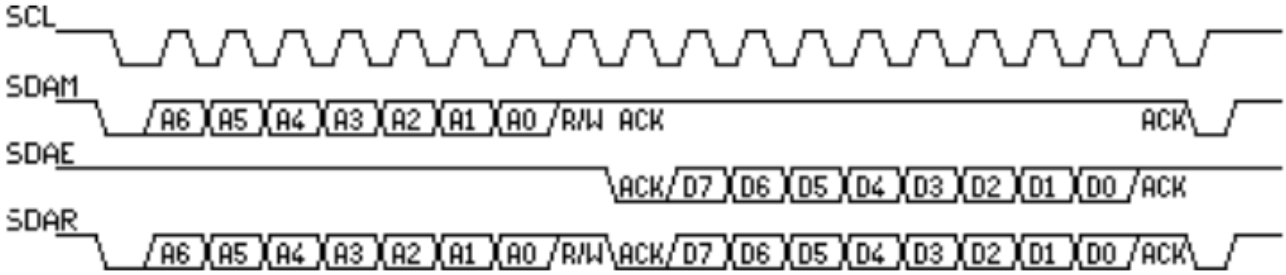


Exemple d'écriture d'une donnée.

Remarque : dans le cas particulier 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'acquittement ACK avant de passer à la suite.

3.5. Lecture d'une donnée

La lecture d'une donnée par le maître se caractérise 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'arrêt.



Exemple de lecture d'une donnée.

4. La gestion des conflits

4.1. 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 grâce l'utilisation de collecteurs ouverts, il faut pouvoir détecter cet état de fait pour éviter la corruption des données transmises.

4.2. Principe

Comme nous l'avons vu précédemment, pour prendre le contrôle du bus, un maître potentiel doit d'abord vérifier que celui-ci soit libre, et qu'une condition d'arrêt ait bien été envoyée depuis au moins 4,7µs. Mais il reste la possibilité que plusieurs maîtres prennent le contrôle du bus simultanément.

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 envoient 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 :

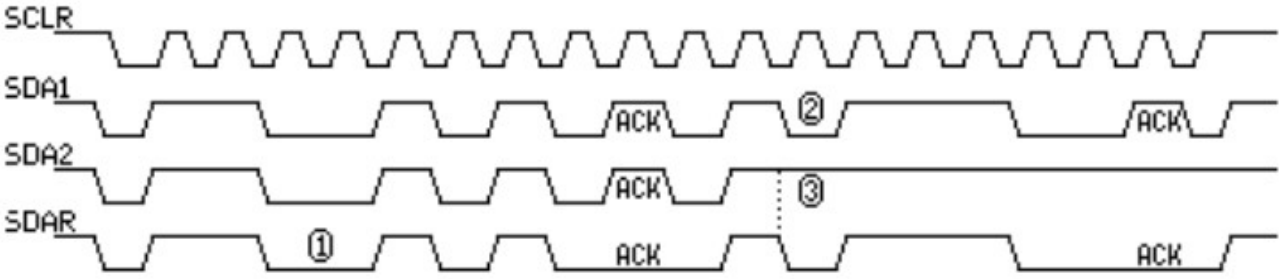
   Il relira forcément '0' et continuera à transmettre. Il ne peut pas alors détecter un éventuel conflit.

3. Un maître 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 contrôle 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.

4.3. Exemple

Soit le chronogramme suivant :



Dans cet exemple :

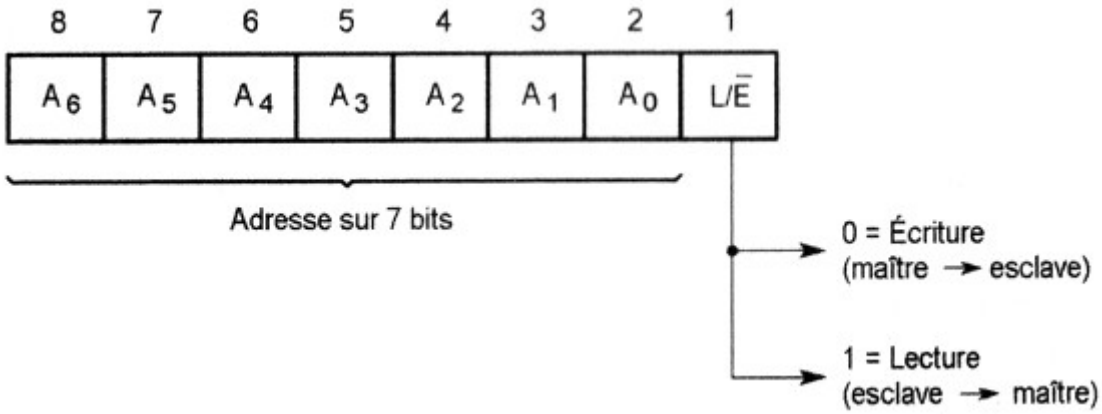
- SCLR : Horloge résultante.
- SDA1 : Niveaux de SDA imposés par le maître n°1.
- SDA2 : Niveaux de SDA imposés par le maître n°2.
- SDAR : Niveaux de SDA réels résultants lus par les deux maîtres.

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.

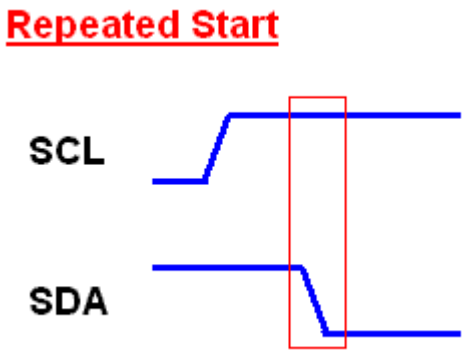
5. En-tête

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.



6. 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).

6.1. Exemple d'écriture du DS1307

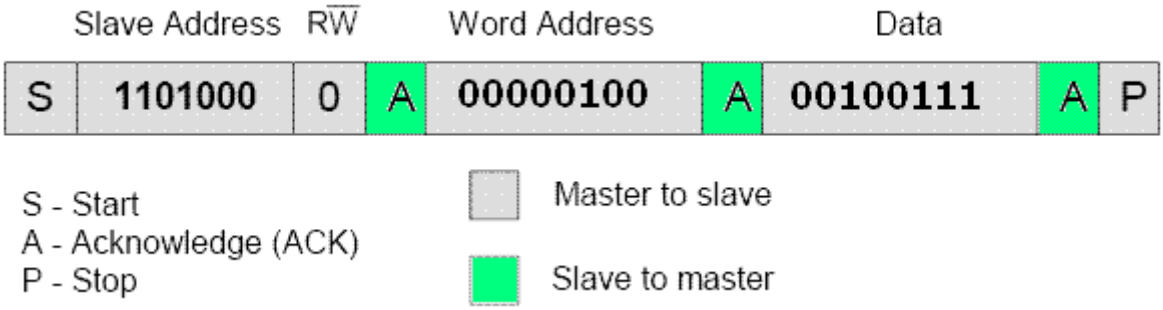
L'émetteur est le maître et le récepteur est l'esclave.

Le registre d'adresse 04h 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

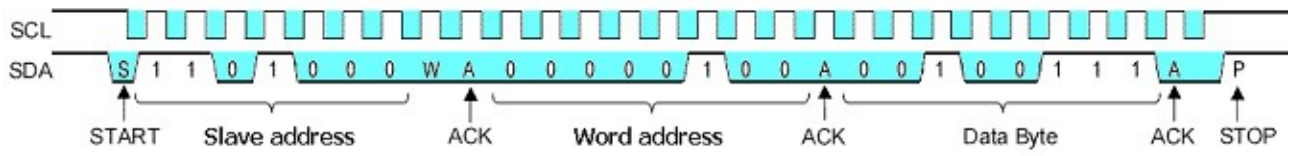
d'adresse 04h du DS1307.

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

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

La fréquence est de 100 kHz, il faut donc pour le transfert :

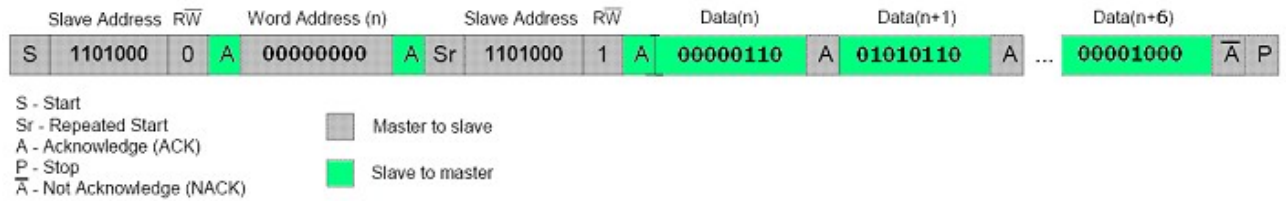
$$\frac{3\times 9}{100000}\approx 270\,ms$$

6.2. Exemple de lecture du DS1307

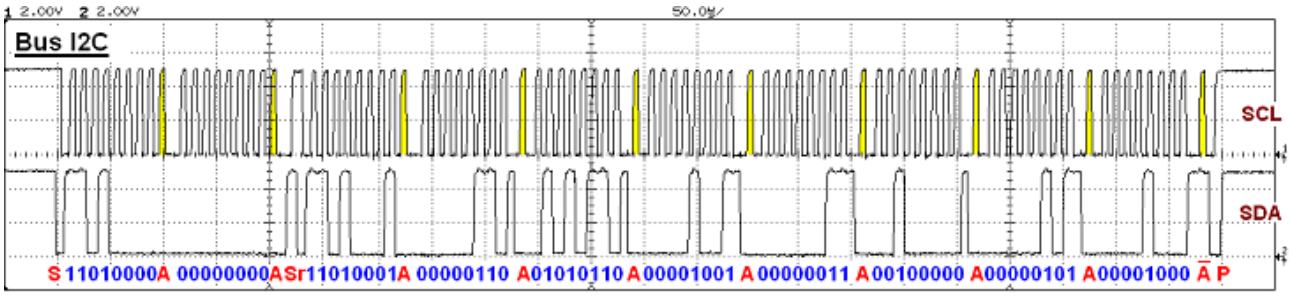
L'émetteur est l'esclave et le récepteur est le maître.

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).

Question : Essayer de retrouver précisément la date et l'heure à l'aide du chronogramme complet de

l'échange sur bus I2C avec l'esclave DS1307.

Le contenu du registre d'adresse 0x00 du DS1307 est 00000110 (codage BCD : 06 secondes).

Le contenu du registre d'adresse 0x01 est 0x56 (c'est-à-dire 56 minutes).

Le contenu du registre d'adresse 0x02 est 0x09 (c'est-à-dire 09 heures).

Le contenu du registre d'adresse 0x03 est 0x03 (c'est-à-dire Mardi).

Remarque : pour les américains Sunday = 1.

Le contenu du registre d'adresse 0x04 est 0x20 (c'est-à-dire 20 ème jour du mois).

Le contenu du registre d'adresse 0x05 est 0x05 (c'est-à-dire mois de mai).

Le contenu du registre d'adresse 0x06 est 0x08 (c'est-à-dire année 2008).

Mardi 20 mai 2008, 9 heures 56 minutes et 6 secondes

7. 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