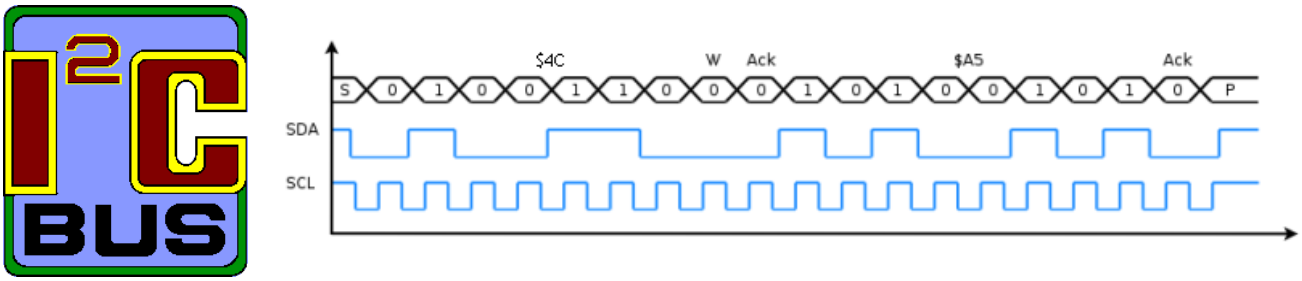
# TRANSMISSION SÉRIE : LE BUS I2C




## Objectifs du COURS :

Ce cours traitera essentiellement les points suivants :

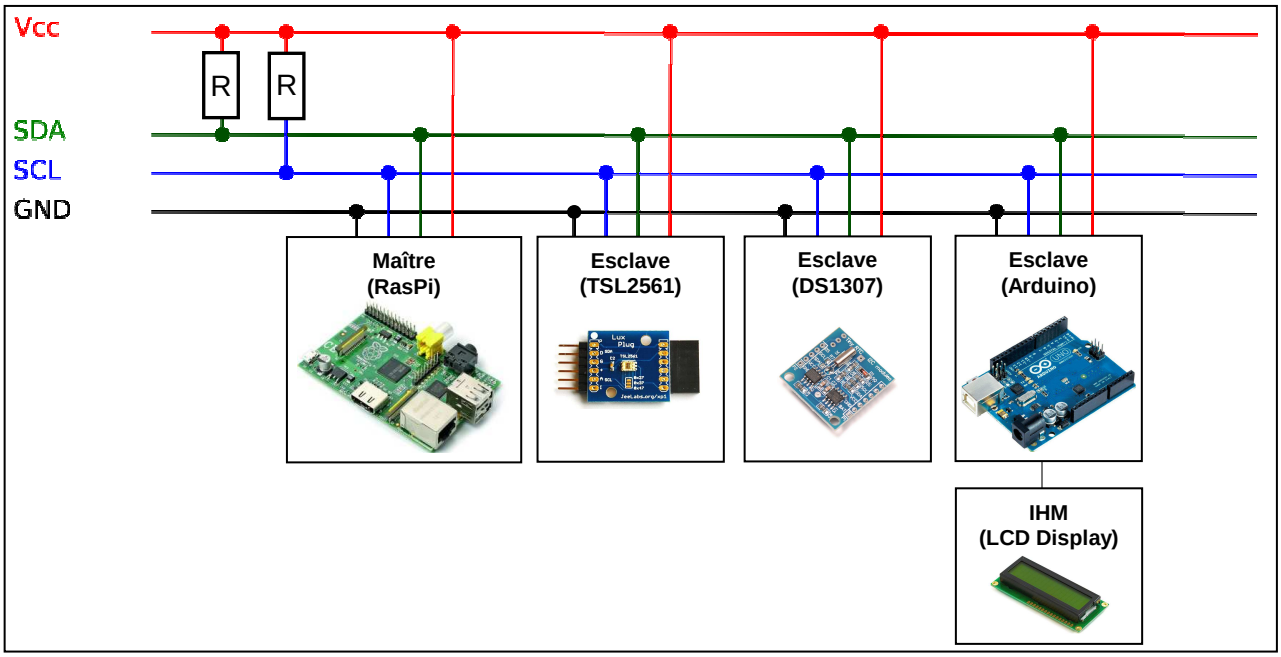
- le bus I2C
  - protocole
  - en-tête
  - exemples avec le circuit Dallas DS1307 (RTC)
- compléments

## 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.

## 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.

## PROTOCOLE

La figure ci-dessous montre le principe adopté au niveau des étages d'entrées/sorties des circuits d'interfaces du bus I2C.

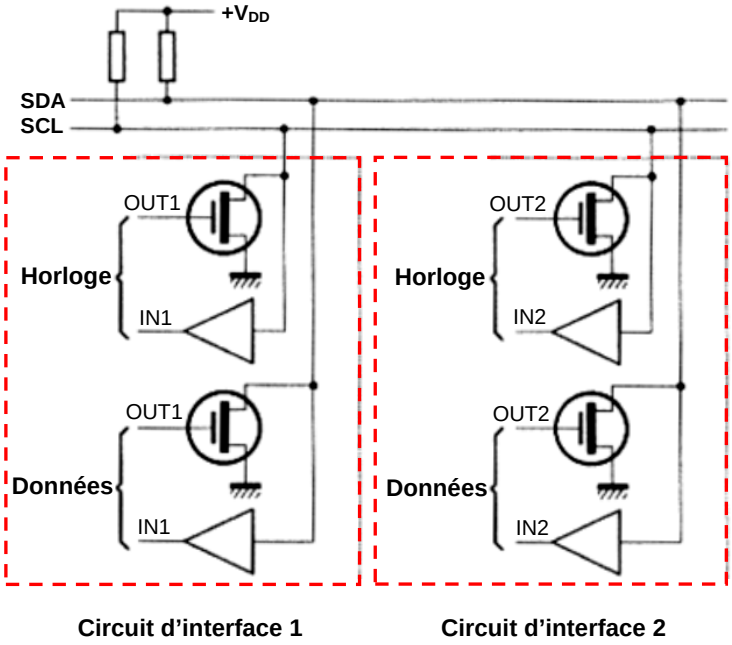
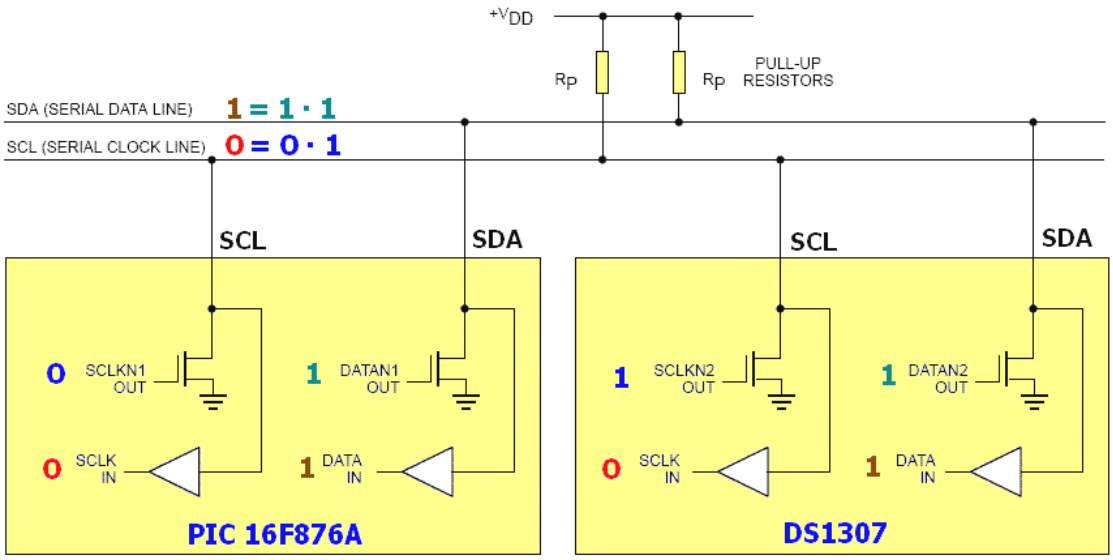



**Schéma de l'interface matérielle des circuits compatibles du bus I2C**

La partie entrée n'appelle aucune remarque particulière, en revanche la partie sortie fait appel à une configuration à drain ouvert (l'équivalent en MOS du classique collecteur ouvert) et qui permet de réaliser des ET câblés par simple connexion sur la ligne SDA ou SCL, des sorties de tous les circuits.
Aucune charge n'étant prévue dans ces derniers, une résistance de rappel (pull-up) à une tension positive doit être mise en place.
Étant entendu que nous travaillons en logique positive, c'est-à-dire qu'un niveau haut correspond à une tension plus élevée qu'un niveau bas.

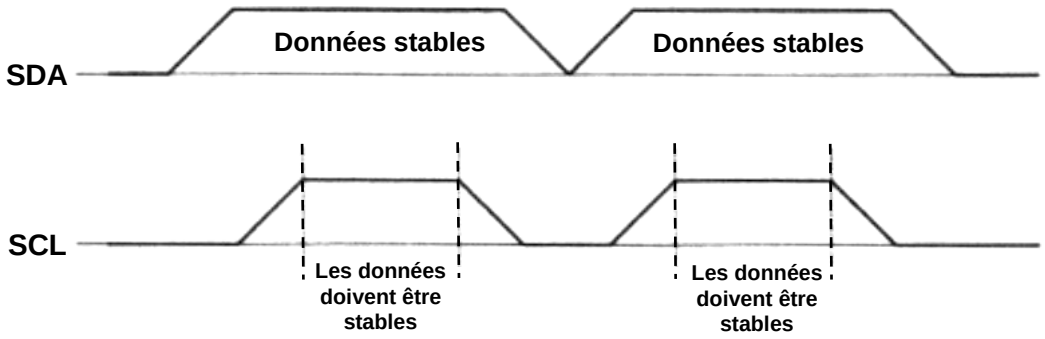
Lorsque aucun abonné n'émet sur le bus, les lignes SDA et SCL sont au niveau haut qui est leur état de repos.
Quand une ligne (SDA ou SCL) est au repos (niveau 1), on peut la forcer à 0.
Quand une ligne (SDA ou SCL) est au niveau 0, on ne peut pas la forcer à 1.




**Exemple avec PIC 16F876A (master) et DS1307 (slave)**

Dans l'exemple ci-dessus, l'horloge SCL est au niveau 0 car c'est le PIC 16F876A (maître) qui force cette ligne à 0.

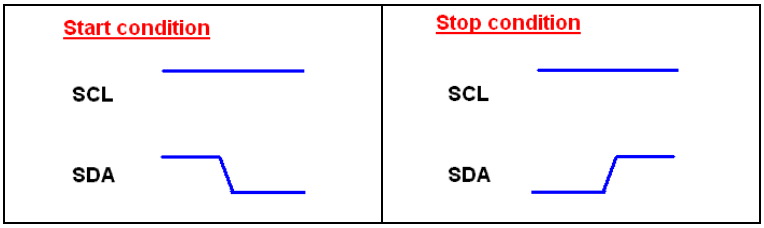
**Remarque :**
Le DS1307 (module RTC : Real Time Clock) fonctionne toujours en esclave : ce n'est donc jamais lui qui agit sur l'état de l'horloge (SCL).




**Chronogramme fondamental du bus I2C**

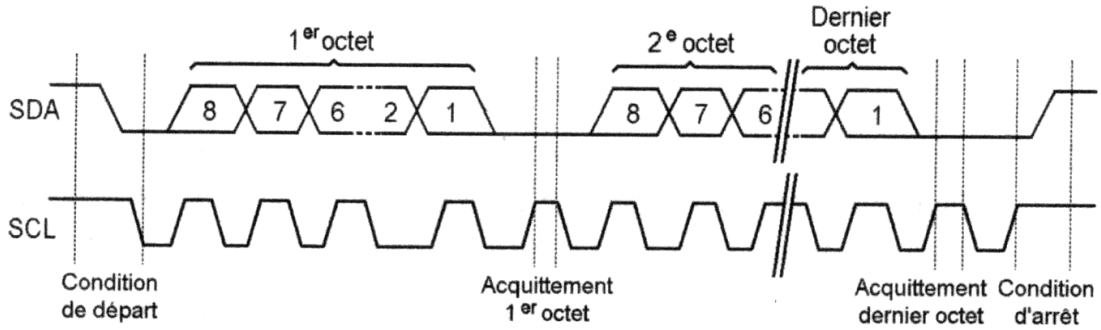
Le chronogramme ci-dessus résume le principe fondamental d'un transfert de données :
Une donnée n'est considérée comme valide sur le bus que lorsque le signal SCL est à l'état haut.
L'émetteur doit donc positionner la donnée à émettre lorsque SCL est à l'état bas et la maintenir tant que SCL est à l'état 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.

Une condition de départ est réalisée lorsque la ligne SDA passe du niveau haut au niveau bas alors que SCL est au niveau haut. Réciproquement, une condition d'arrêt est réalisée lorsque SDA passe du niveau bas au niveau haut alors que SCL est au niveau haut.




Les données sont envoyées par paquets de huit bits (ou octets). Le bit de poids fort est envoyé le premier, chaque octet est suivi par un bit d'acquittement (ACK) de la part du destinataire.

Le processus fonctionne de la façon ci-dessous.

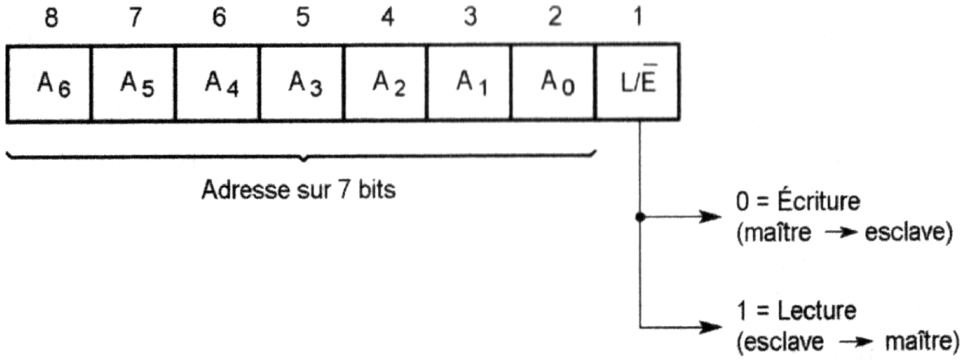



**Chronogramme d'un échange sur bus I2C**

La ligne SCL est pilotée par l'initiateur de l'échange ou maître.
Le chronogramme ci-dessus montre d'abord une condition de départ, généré par le maître du bus à cet instant. Elle est suivie par le premier octet de données, poids fort en tête. Après le huitième bit, l'émetteur qui est aussi le maître dans ce cas met sa ligne SDA au niveau haut, c'est-à-dire au repos mais continue à générer l'horloge sur SCL.
Pour acquérir l'octet, le récepteur doit alors forcer la ligne SDA au niveau bas pendant l'état haut de SCL qui correspond à cet acquittement, prenant en quelque sorte la place d'un neuvième bit.
Le processus peut alors continuer avec l'octet suivant et se répéter autant de fois que nécessaire pour réaliser un échange d'informations complexes.
Lorsque cet échange est terminé, le maître génére une condition d'arrêt.

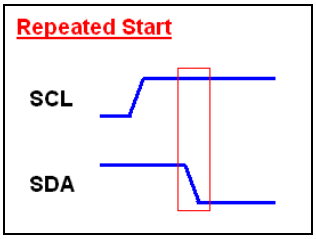
## EN-TÊTE

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.

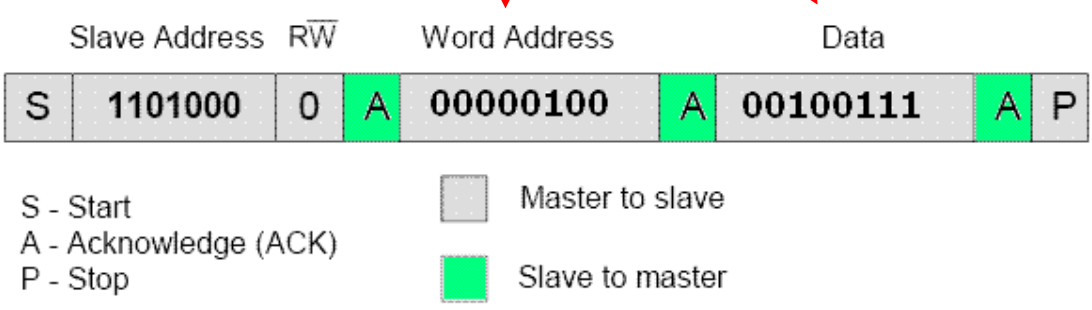



## EXEMPLE AVEC L'ESCLAVE DS1307

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).

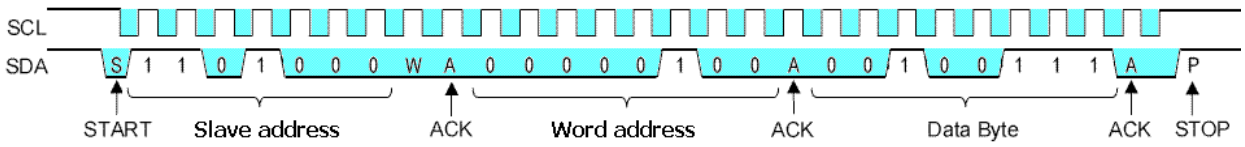
**Exemple d'écriture du DS1307 :**
L'émetteur est le maître et le récepteur est l'esclave.
Le registre d'adresse 04h du DS1307 contient la date (voir datasheet du DS1307).
Pour régler le calendrier au 27 du mois, il faut écrire la donnée 27 (en code BCD) dans le registre d'adresse 04h du DS1307.

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 :***

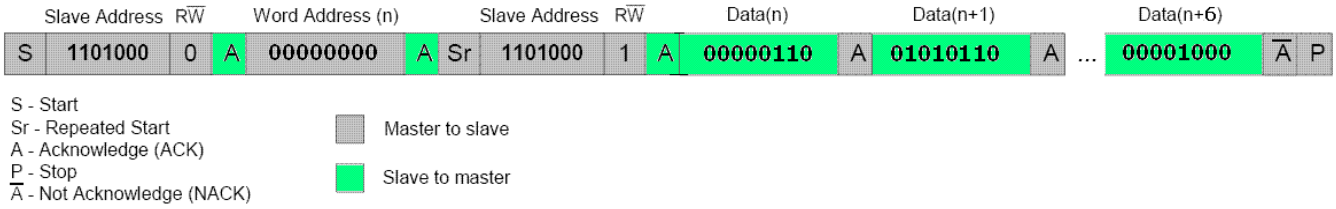
*D'après l'exemple ci-dessus, combien de temps faut-il pour le transfert en écriture ?*

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

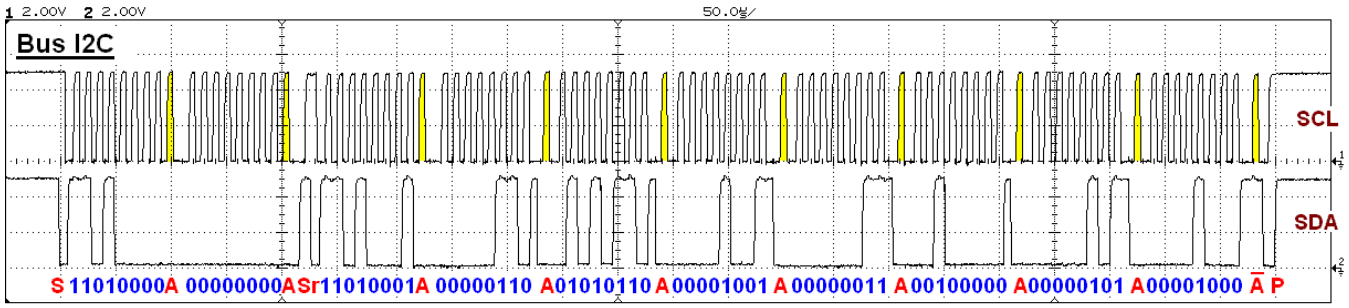
$\frac{3x9}{100000}$ **= 270 µs environ.**

**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 (page 7).*

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

## 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