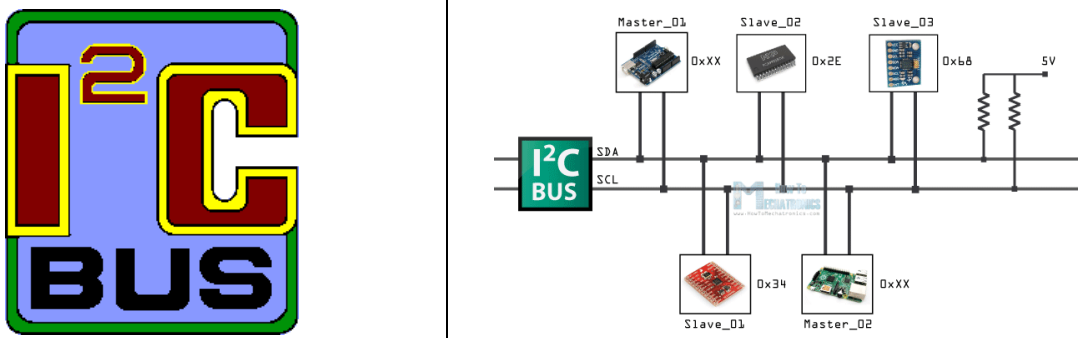


Objectifs du cours :

Ce cours traitera essentiellement les points suivants :

- Le bus I2C :
  - protocole
  - en-tête
- Exemples avec le circuit Dallas DS1307 (RTC)
- La gestion des conflits
- Compléments (Wire Library pour Arduino, les adresses réservées, caractéristiques électriques)
- Exercices d'application

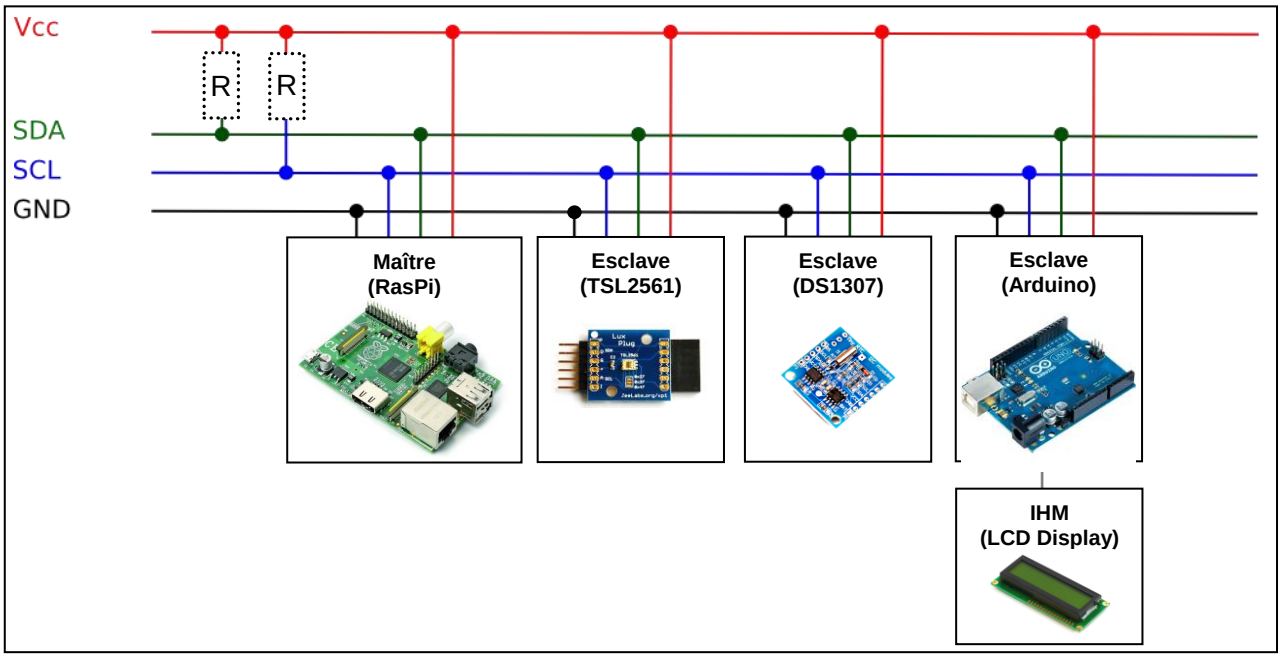
INTRODUCTION

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 (Uno) 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).
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 (la solution est matérielle ou logicielle).

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.
Sur la figure (page précédente) les pull-up sont représentées en pointillées, elles ne sont pas nécessaires pour l'Arduino et le Raspberry car les résistances de tirage sont intégrées sur les cartes.
Le bus I2C permet d'échanger des informations sous forme série avec un débit pouvant atteindre 100 kbps (standard mode), 400 kbps (fast mode) ou 3,2 Mbps (high speed mode) pour les versions les plus récentes.

Ses points forts sont les suivants :
- c'est un bus série bifilaire (8 bits) 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.

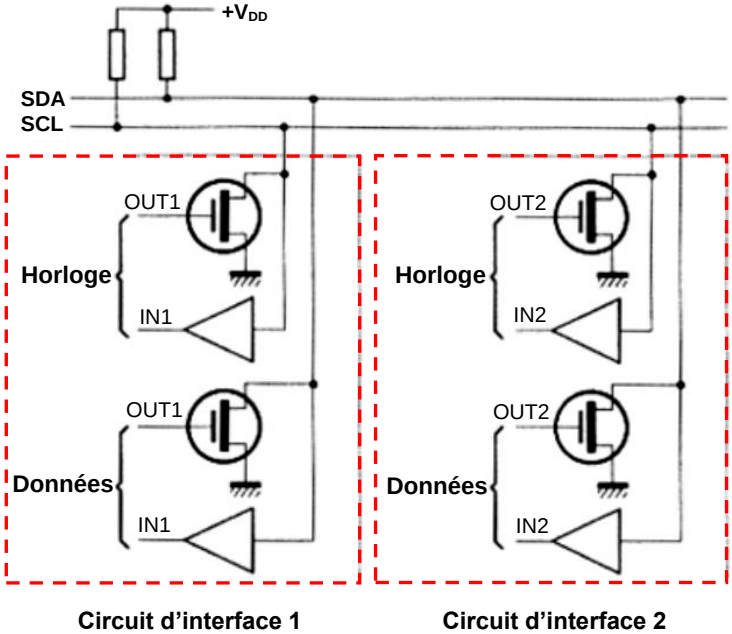
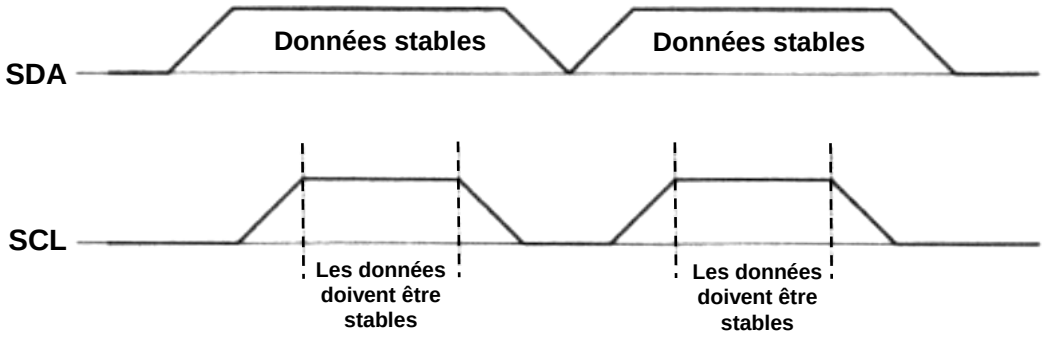


Schéma de l'interface matérielle des circuits compatibles du bus I2C

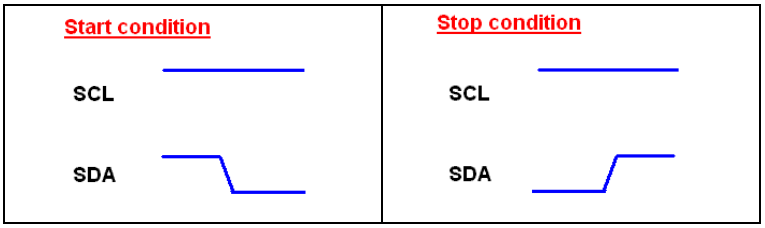
La partie entrée n'appelle aucune remarque particulière, en revanche la partie sortie fait appel à une configuration à drain ouvert (l'équivalent en MOS du classique collecteur ouvert) et qui permet de réaliser des ET câblés par simple connexion sur la ligne SDA ou SCL, des sorties de tous les circuits.
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.
Lorsqu'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.

CHRONOGRAMME FONDAMENTAL



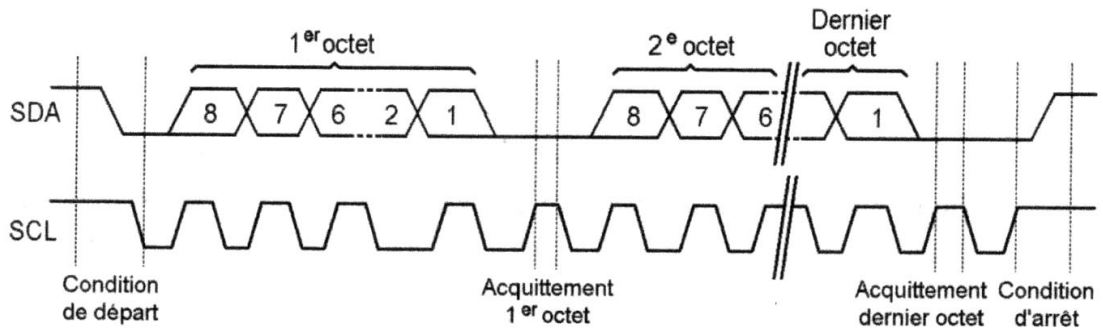
Chronogramme fondamental du bus I2C

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.

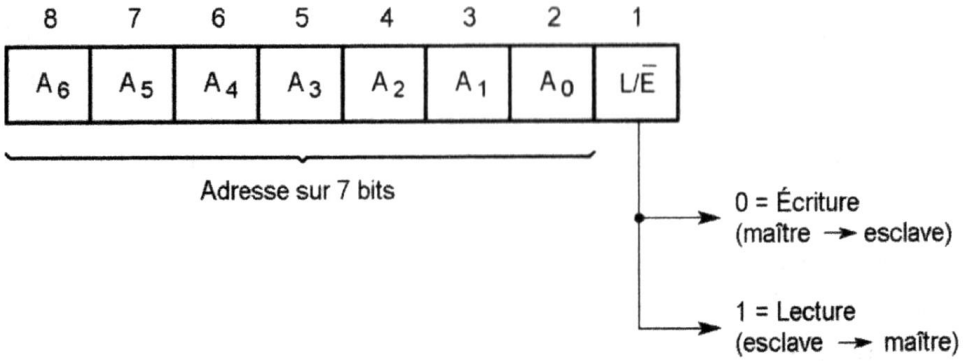


Chronogramme d'un échange sur bus I2C

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 génère une condition d'arrêt.

EN-TÊTE

La figure ci-dessous 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.

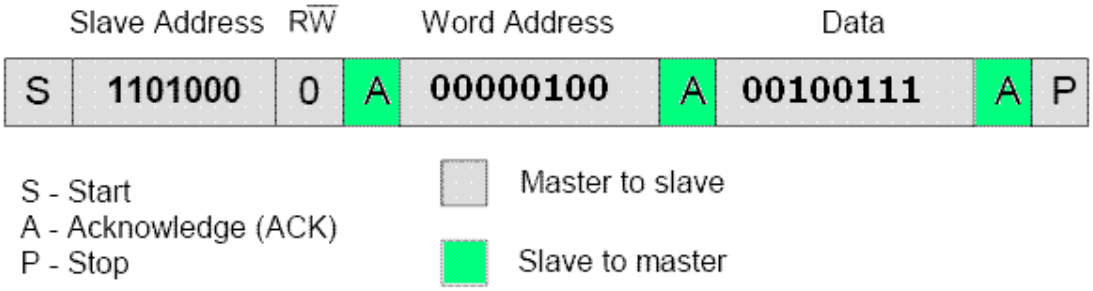
EXEMPLES AVEC LE CIRCUIT DALLAS DS1307 (RTC)

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 avec un débit binaire dans le mode standard.
L'adresse I2C (7 bits) du DS1307 est : 1101000 (adresse fixée par le constructeur et non modifiable).

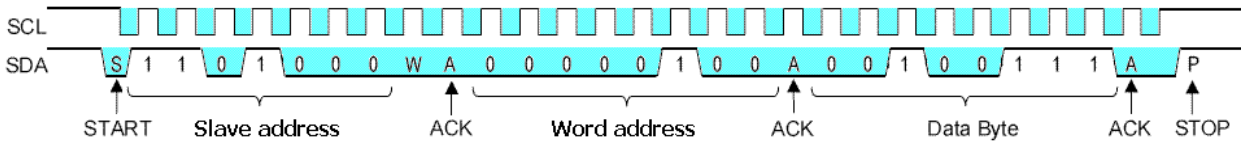
EXEMPLE D'ÉCRITURE

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

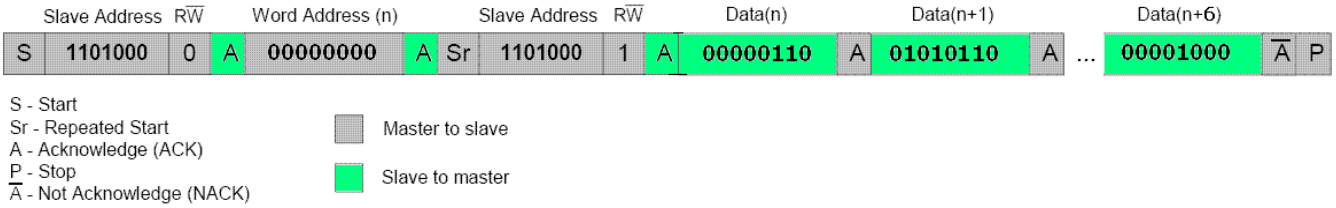
Question

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

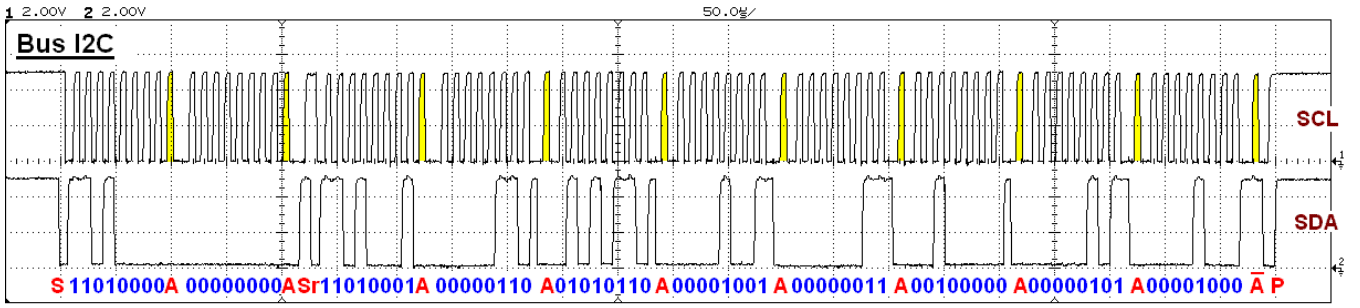
……...

EXEMPLE DE LECTURE

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).

……

……

……

……

……

……

……

LA GESTION DES CONFLITS

La structure du bus I2C a été conçue pour pouvoir y accueillir plusieurs maîtres. Se pose alors le problème commun à tous 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 un conflit logique et donc éviter une collision entraînant la corruption des données transmises.

PRINCIPE

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 :

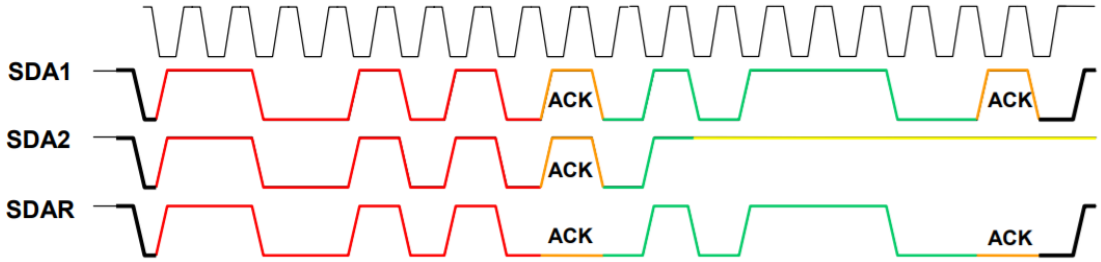
① Les différents maîtres envoient les mêmes données au même moment :
Les données ne sont pas corrompues, la transmission s'effectue normalement, comme si un seul maître avait parlé. Ce cas est rare.

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 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 contrôle du bus, pour ne pas perturber la transmission du second. Il continue néanmoins à lire les données au cas où celles-ci lui auraient été destinées.

Exemple :

Soit le chronogramme ci-dessous.



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

Le premier octet est transmis normalement car les deux maîtres imposent les mêmes données. 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. 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.
Au total, l'esclave a reçu les données du maître n°1 sans erreurs et le conflit est passé inaperçu.

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

EXERCICES D'APPLICATION

EXERCICES N°1

Question 1

D'après le circuit page suivante, DEV1 désire prendre la parole (START). Avant de pouvoir émettre, ***il doit s'assurer que le bus est libre, en effectuant un contrôle de l'état logique des signaux SDA et SCL.*** *Donnez l'équation logique de EMISSION = f(SCL IN, SDA IN) autorisant l'émission.*

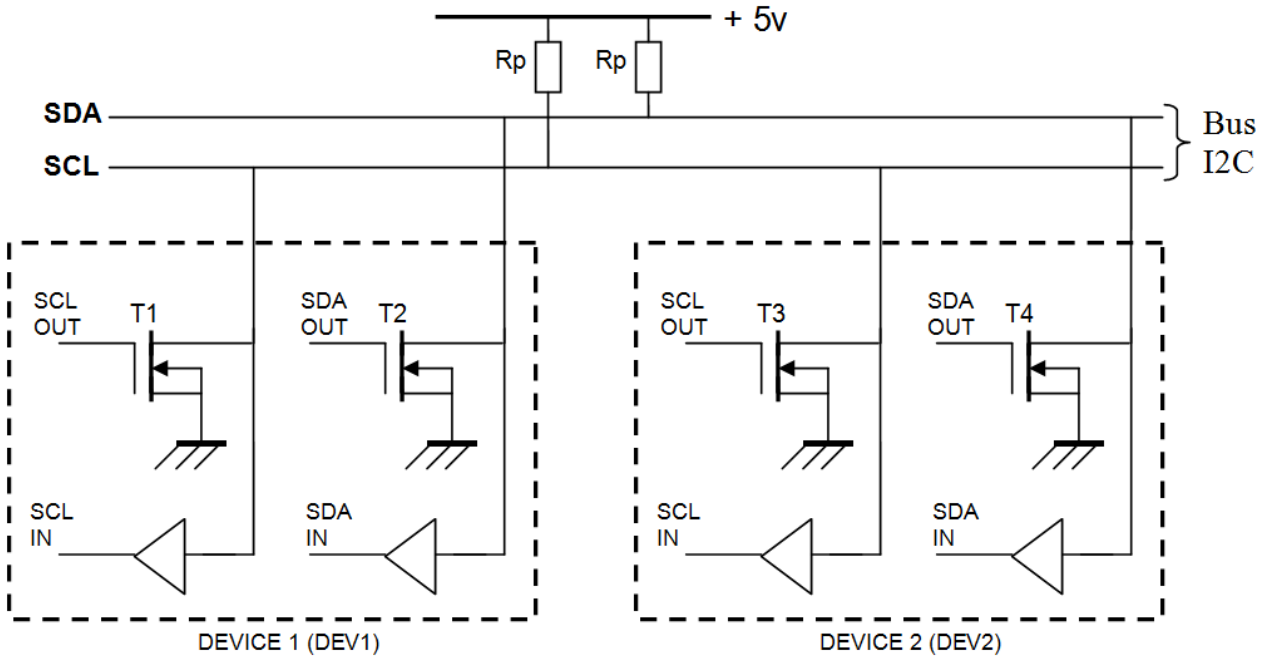
……..

Question 2

Lors d'une prise de parole du circuit DEV1 (START), indiquez, dans l'ordre chronologique, l'état des transistors T1 et T2.

……..

……



Question 3

Dans quel état doit se trouver T2 afin de signifier au destinataire de la donnée, que le dernier bit a été transmis ?

……

Question 4

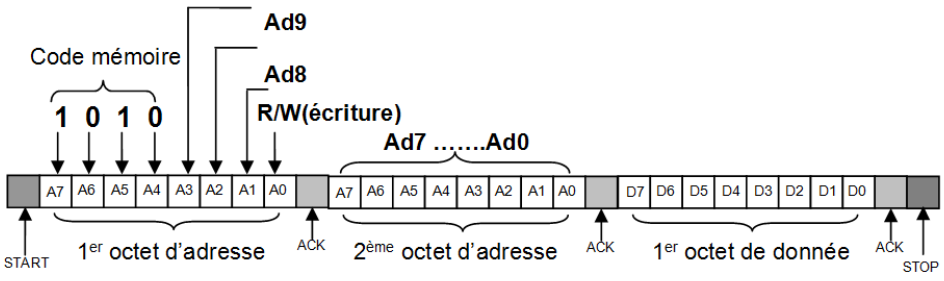
Le circuit DEV1 désire clôturer la prise de parole (STOP). Indiquez, dans l'ordre chronologique, l'état des transistors T1 et T2.

……

……

Question 5

Dans le cas d'un circuit nouvelle génération haut débit (3,2 Mbps), quel temps mettra un circuit pour transmettre un octet (hors temps de START et de STOP) selon le schéma ci-dessous ?



……..

……..

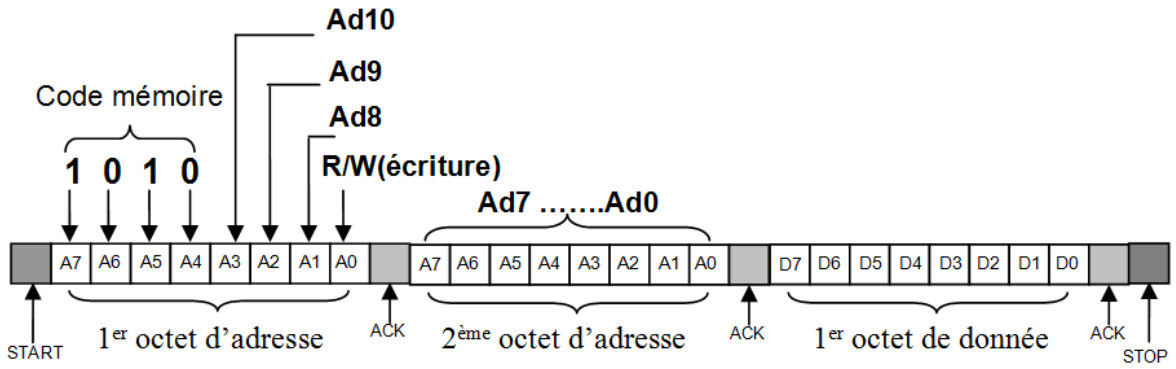
EXERCICES N°2

Lorsque l'adressage de la mémoire nécessite plus de 7 bits d'adresse, on distingue les 8 bits de la partie basse de l'adresse (Ad7 à AD0) et la partie haute comprenant 3 bits (Ad8 à Ad10).

Le protocole I2C fait appel à un code spécifique (1010) lorsqu'il s'agit de communiquer avec une mémoire compatible I2C.

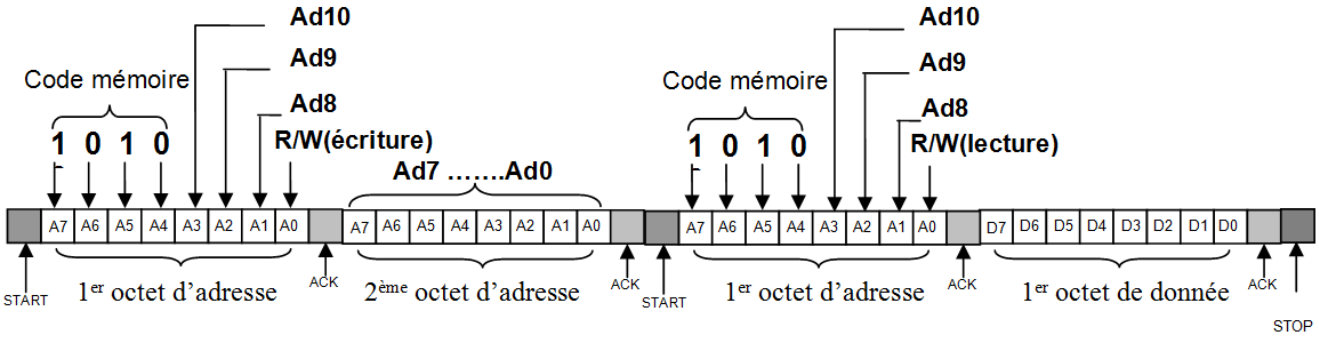
Principe d'écriture dans une mémoire avec adressage effectué sur 2 octets :

Le maître pointe l'adresse de la mémoire (2 octets) en mode écriture,
puis écrit la donnée (1 octet).



Principe de lecture dans une mémoire avec adressage effectué sur 2 octets :

Le maître commence par pointer l'adresse de la mémoire (2 octets) en **mode écriture**, puis pointe la partie haute de l'adresse (1 octet) en mode lecture, avant de lire la donnée (1 octet). Lors de la réception de la donnée, c'est le maître qui devra accuser réception de la donnée en envoyant à la mémoire, un bit ACK.

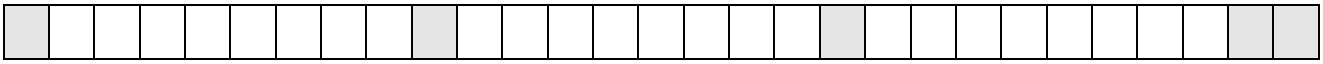


Question 1

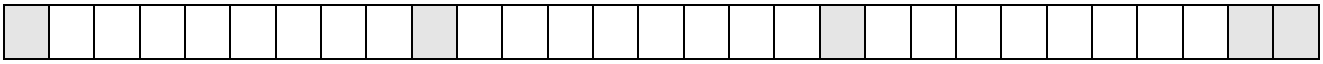
On désire écrire les données suivantes : $A7 et $5C dans les mémoires respectivement d'adresse $01F et $5D1.

*Établissez les trames (en binaire) des informations que le maître devra envoyer sur la ligne **SDA** du bus I2C. Répondez en complétant les trames ci-dessous.*

Écriture : adresse $01F, donnée $A7



Écriture : adresse $5D1, donnée $5C



Question 2

On désire lire la donnée $8D stockée à l'adresse $2FA.
*Établissez la trame (en binaire) des informations que le maître devra envoyer sur la ligne **SDA** du bus I2C. Répondez en complétant la trame ci-dessous.*

Lecture : donnée $8D stockée à l'adresse $2FA