

E3020 Systèmes embarqués

CM3-BUS DE COMMUNICATION + I2C CONCEPTS

Objectifs

1. Pouvoir caractériser un bus de communication
2. Avoir un aperçu des performances des différents bus
3. Expliquer les particularités du bus I2C
 1. Interface
 2. Arbitrage
 3. Adressage
 4. Trame

Bus de communication

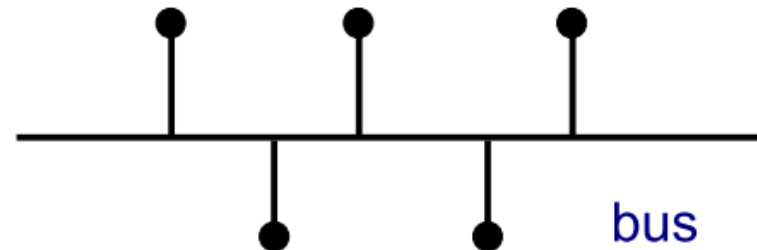
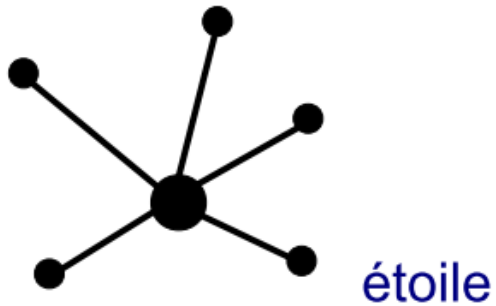
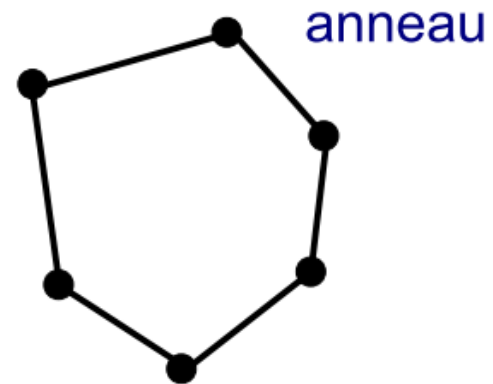
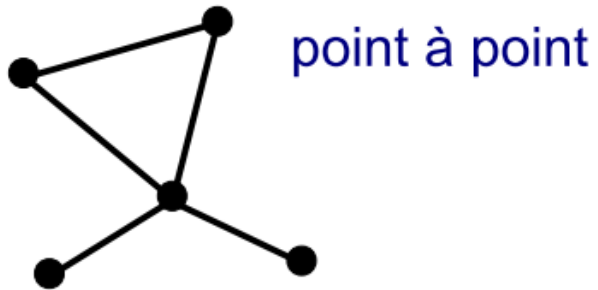
Qu'est-ce qu'un bus de communication ?

A quoi sert-il?

Quels sont les caractéristiques d'un bus?



Bus de communication



Bus de communication

Un bus est une topologie de réseau où tous les nœuds sont à un seul lien, appelé le bus. Ce bus peut contenir plusieurs lignes, toutes communes à chaque nœud du réseau.

De ce fait, toute donnée émise sur un bus est reçue par tous les nœuds en même temps. Pour autant que le lien physique composant le bus ne soit pas trop long.

Bus de communication

- Le rôle fondamental est d'assurer une communication entre le microcontrôleur et d'autres circuits intégrés qui lui sont connectés.
- Tous les bus ne sont pas implémentés sur tous les microcontrôleurs. Il s'agit d'un point important pour le choix du modèle.
- Le choix du bus se fait par les caractéristiques de ce dernier

Bus de communication





- Un bus de communication se partage entre plusieurs puces. Une puce pour pouvoir communiquer avec UNE autre, doit l'atteindre via son **adresse**. Il existe des moyens pour toucher toutes les puces d'un coup en utilisant des adresses réservées.
- La communication sert à transmettre une information qui sera enregistrée dans un registre. Il faudra donc également préciser son **adresse dans la puce**.
- Enfin, **la donnée** sera transmise.
- Ces trois paramètres sont quasi universels. C'est le protocole et la connectique qui distingue les bus entre eux (nombre de fil et trame à suivre)

Bus de communication

- Donnez des noms de protocole de communication utilisant une topologie de bus?



Quelques définitions :

1. Série/parallèle 
2. Relation maître/esclave (voire multi-maîtres) 
3. Synchrone/asynchrone 
4. Half-duplex/full-duplex 

Caractéristiques


Nombre de puce connectables ?

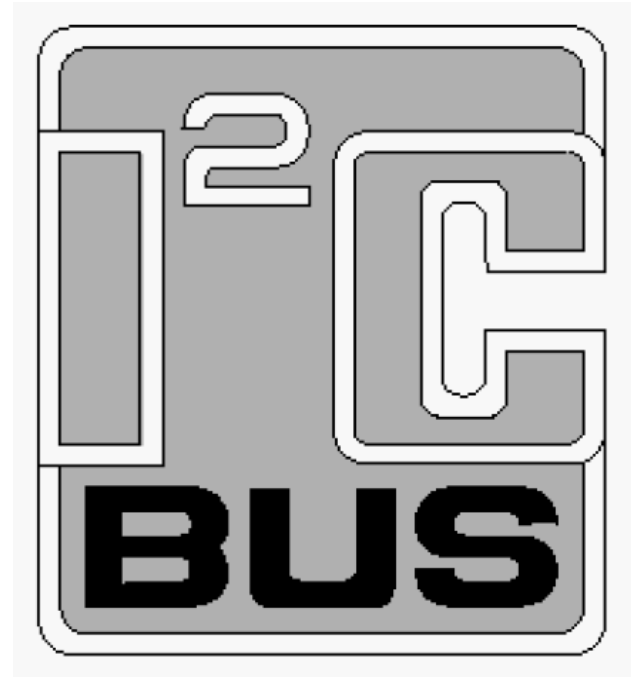


Inter-Integrated Circuit (I2C)

- Il s'agit d'un bus série de bas niveau développé par Philips Semiconductor en 1982 pour les applications de domotique.
- Il est de courte portée (quelques mètres)
- Fonctionne généralement à 100kb/s (nouvelle génération à 3Mb/s).
- Il utilise deux lignes (en plus de la masse)

Inter-Integrated Circuit (I2C)


- Caractéristiques :
 - Communication série
 - Synchrone 
 - Maître-esclave
 - Half-duplex
 - Accepte le multi-maître
 - Adressage sur 7 bits (voire mode 10 bits), donc entre 128 et 1024 éléments connectables. Cependant certaines adresses sont réservées pour le broadcast. En pratique, les constructeurs ont tendance à fixer l'adresse de certains nœuds. Cela peut causer des collisions.



Serial Peripheral Interface (SPI)

- Développé par Motorola pour le microcontrôleur 68xx
- Les nœuds du réseau partagent trois fils en commun (en plus de la masse). Il faut ajouter un fil entre chaque esclave et le maître.
- Un maître unique! Il n'est pas possible d'avoir plusieurs maîtres sur le réseau.
- A chaque nœud esclave est ajouté une ligne SS (slave select).
Exception où l'adressage

Serial Peripheral Interface (SPI)

- Caractéristiques :
 - Maître/esclaves!
 - Synchrone 
 - Full duplex car maître et esclave peuvent transmettre simultanément.
 - Sa portée est de quelques mètres
 - Son débit est plus élevé que l'I2C. Il est possible d'atteindre 100Mbits/s.

Serial Peripheral Interface (SPI)

AVANTAGES

Flexibilité du nombre de bits à transmettre ainsi que le protocole

Simplicité de l'interface matérielle

- Aucun arbitre nécessaire car aucune collision possible
- Les esclaves utilisent l'horloge du maître et n'ont donc pas besoin d'oscillateur propre.

INCONVÉNIENTS

Pas de contrôle physique de données, ni d'acquittement

Monopolise plus de broches (pin)

Aucun adressage possible

Un seul maître (sauf exception rare)

Distance relativement courte

RS-232 (UART/USART)

Il s'agit d'un bus conçu en 1962.

Il est possible de l'utiliser sur plusieurs dizaine de mètres. Cette caractéristique de réduire le débit atteignable : entre 2400 et 115200b/s.

Historiquement, il était utilisé pour faire la liaison entre l'ordinateur et le port série.

Le bus fonctionne au moins avec deux lignes (plus la masse) : Tx et Rx.

D'autres lignes sont disponibles pour améliorer la fiabilité de la liaison mais elles ne sont pas obligatoires.

RS-232 (UART/USART)

Caractéristiques :

- Ne permet de lier que deux périphériques entre eux en mode duplex car les lignes d'émission et de réception sont croisés : Rx1 vers Tx2 et Rx2 vers Tx1.
- De façon synchrone ou asynchrone : c'est au logiciel de gérer comment se déroule la communication.
- Full duplex
- Bus de choix pour les petits projets.

| Signal | | Origin | | DB-25 | DE-9 |
|---------------------|-------------|--------|-----|-------|------|
| Name | Abréviation | DTE | DCE | | |
| Transmitted Data | TxD | ● | | 2 | 3 |
| Received Data | RxD | | ● | 3 | 2 |
| Data Terminal Ready | DTR | ● | | 20 | 4 |
| Data Carrier Detect | DCD | | ● | 8 | 1 |
| Data Set Ready | DSR | | ● | 6 | 6 |
| Ring Indicator | RI | | ● | 22 | 9 |
| Request To Send | RTS | ● | | 4 | 7 |
| Clear To Send | CTS | | ● | 5 | 8 |
| Signal Ground | G | common | | 7 | 5 |
| Protective Ground | PG | common | | 1 | NC |

1 Wire

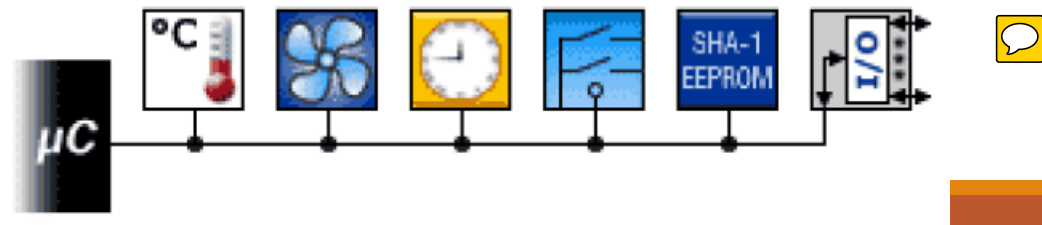
Mis au point par Dallas Semiconductor Corp.

Fonctionne avec un seul fil et une masse.

Portée est de plusieurs centaines de mètres. Le débit faible est de 16,3kbit/sec.

Les nœuds sont adressés sur 64bits (ce qui donne 2^{64} adresses possibles). Une commande d'énumération permet de faire un appel pour détecter les nœuds d'un réseau rapidement.

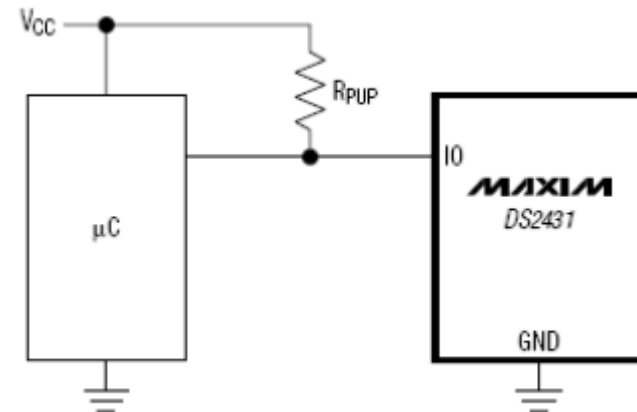
Il s'agit d'un protocole propriétaire. Les implémentations du nœud maître est libre mais pas les nœuds esclaves. La société décide des périphériques utilisables sur son bus.



1 Wire

Caractéristiques :

- Un maître et plusieurs esclaves
- Asynchrone
- Half-duplex
- Coûte moins cher que l'I2C
- Bidirectionnel
- Possibilité d'alimenter les esclaves

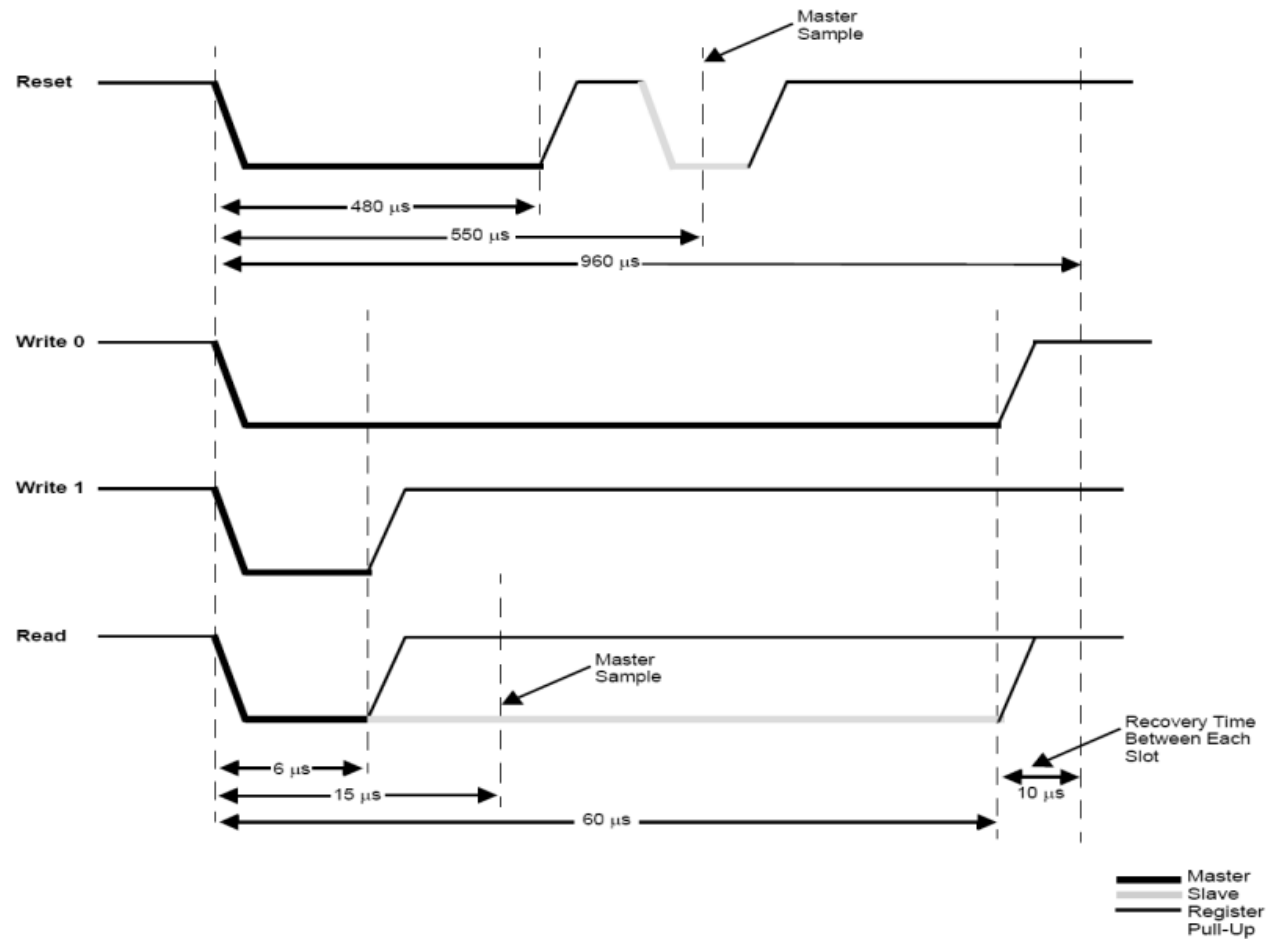


1-Wire

4 opérations possibles : Reset / Write 1 / Write 0 / Read

| Operation | Description | Implementation |
|-------------|--|--|
| Reset | Reset the 1-Wire bus slave devices and get them ready for a command. | Drive bus low, delay 480 μ s. Release bus, delay 70 μ s. Sample bus: 0 = device(s) present, 1 = no device present Delay 410 μ s. |
| Write 0 bit | Send '0' bit to the 1-Wire slaves (Write 0 slot time). | Drive bus low, delay 60 μ s. Release bus, delay 10 μ s. |
| Write 1 bit | Send '1' bit to the 1-Wire slaves (Write 1 slot time). | Drive bus low, delay 6 μ s. Release bus, delay 64 μ s. |
| Read bit | Read a bit from the 1-Wire slaves (Read time slot). | Drive bus low, delay 6 μ s. Release bus, delay 9 μ s. Sample bus to read bit from slave. Delay 55 μ s. |

1-Wire



Universal Serial Bus (USB)

Apparue en 1996.

On ne l'utilise pas à plus de 5mètres.

Vitesse de transfert de l'ordre du Gbit (USB3)

Utilise une paire différentielle : 

Minimum 4 fils :

1. GND
2. D-
3. D+
4. VCC

Entre pairs, synchrone/asynchrone, half-duplex, protocole complexe.



Limitations physiques

1. Temps de propagation dépend de la longueur et de la qualité du câble
2. Atténuation du signal
3. Induction entre les câbles
4. Interférences externes (champs électromagnétique, hachage,...)

Bus de communication

I2C – INTER-INTEGRATED CIRCUIT

Bus : I2C

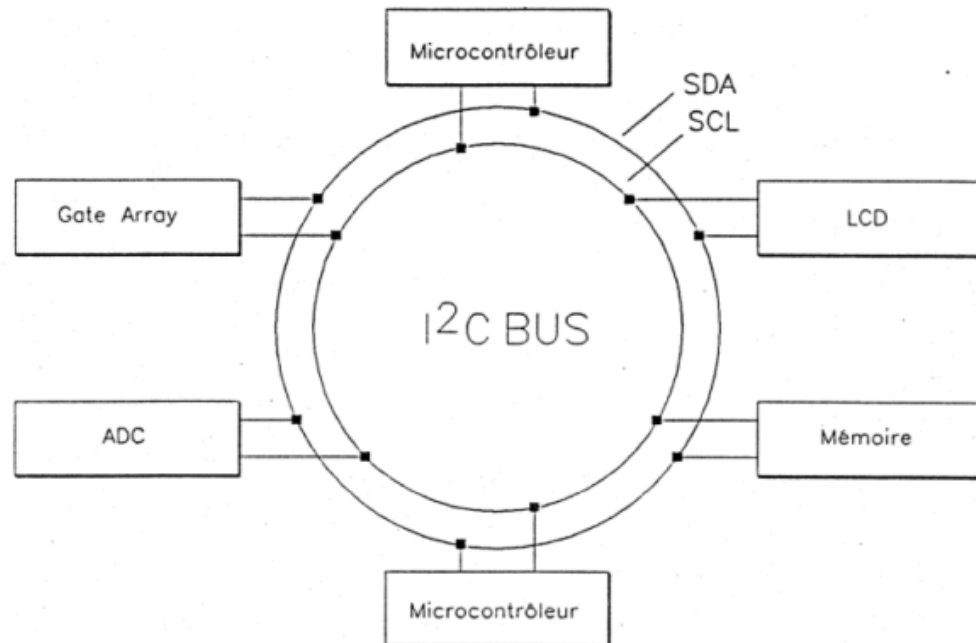
- **Caractéristiques :**

- **Série** : les données sont envoyées les unes derrière les autres.
- **Synchrone** : la communication est régie par une clock.
- **Maître-esclave** : Un communicant domine l'autre.
- **Half-duplex** : On envoie ou on reçoit des données. Mais pas les deux en même temps.

Bus : I2C - Câblage

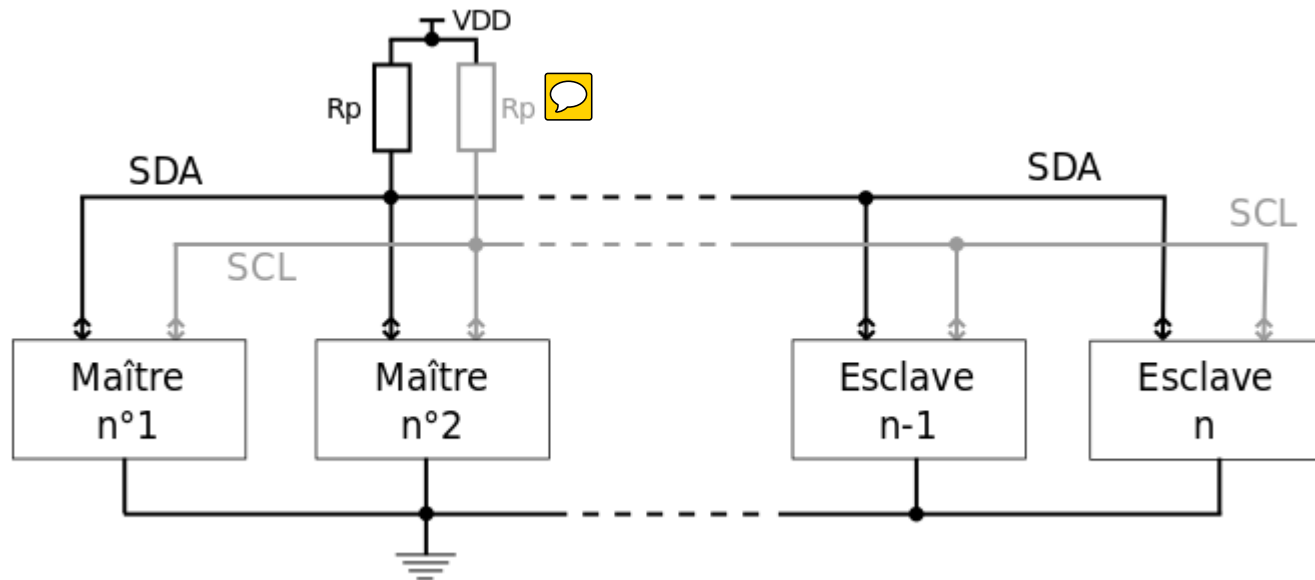
- Deux connexions sont nécessaires :

- **SDA** (serial Data) est la connexion par laquelle passera la donnée.
- **SCL** (serial clock) donne la référence de clock de la transmission. Il s'agit de la fréquence à laquelle les bits se suivent sur SDA.



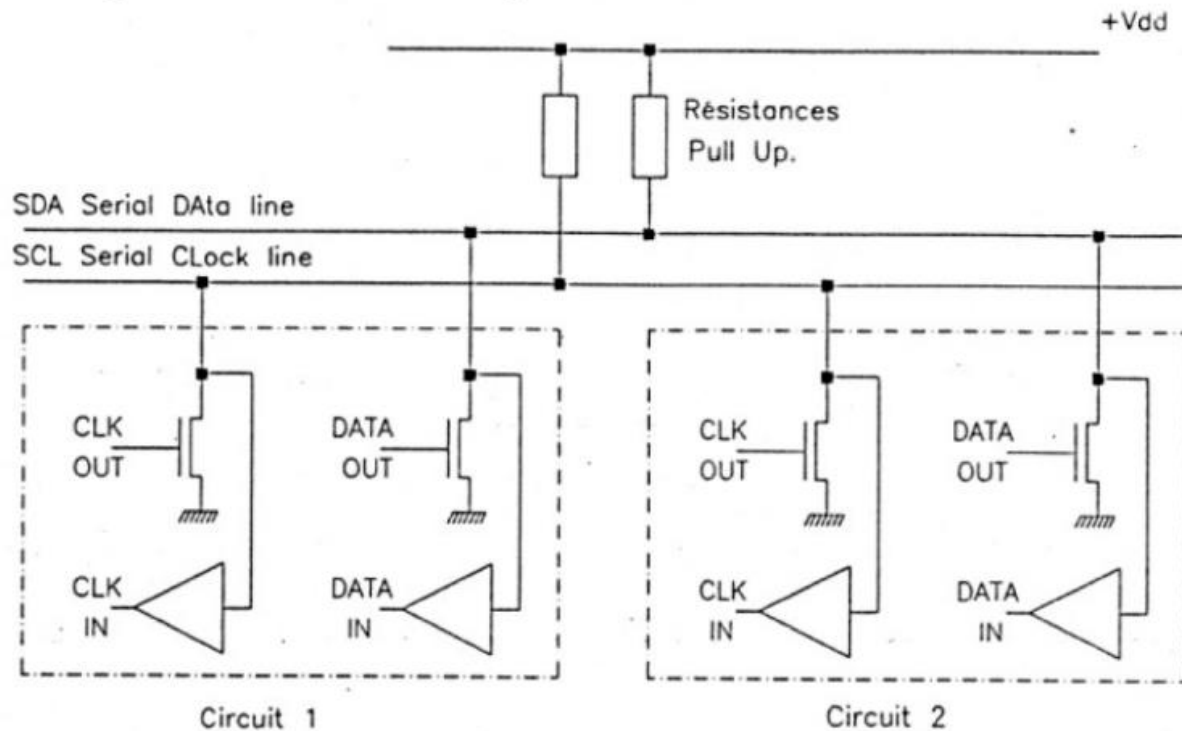
Bus : I2C - Câblage

- Deux connexions sont nécessaires :
 - Ces deux connexions sont partagées par l'ensemble des puces qui communiquent en I2C.
 - Dans cette connexion, il y a un maître qui impose la clock (SCL). SDA peut être utilisé par toutes les puces.



Bus : I2C - Communication

- Une puce (maître ou esclave) pour communiquer va imposer la valeur du bit sur SDA.
- Voici l'étage de sortie de la puce. Comment cela fonctionne-t-il?



Bus : I2C - Communication

- L'étage de sortie est réalisé avec un NMOS en source commun.
- Cette configuration est pratique pour imposer un zéro sur SDA, mais pas pour imposer un 'un'. Il faut donc ajouter une résistance de pull-up. Par défaut, SDA est donc à '1'
- C'est le même schéma pour SCL mais seul le maître peut imposer la clock.
- Quand la puce reçoit l'information, il y a un buffer supplémentaire pour éviter de consommer trop de courant et donc faire chuter la tension sur la résistance de pull-up.

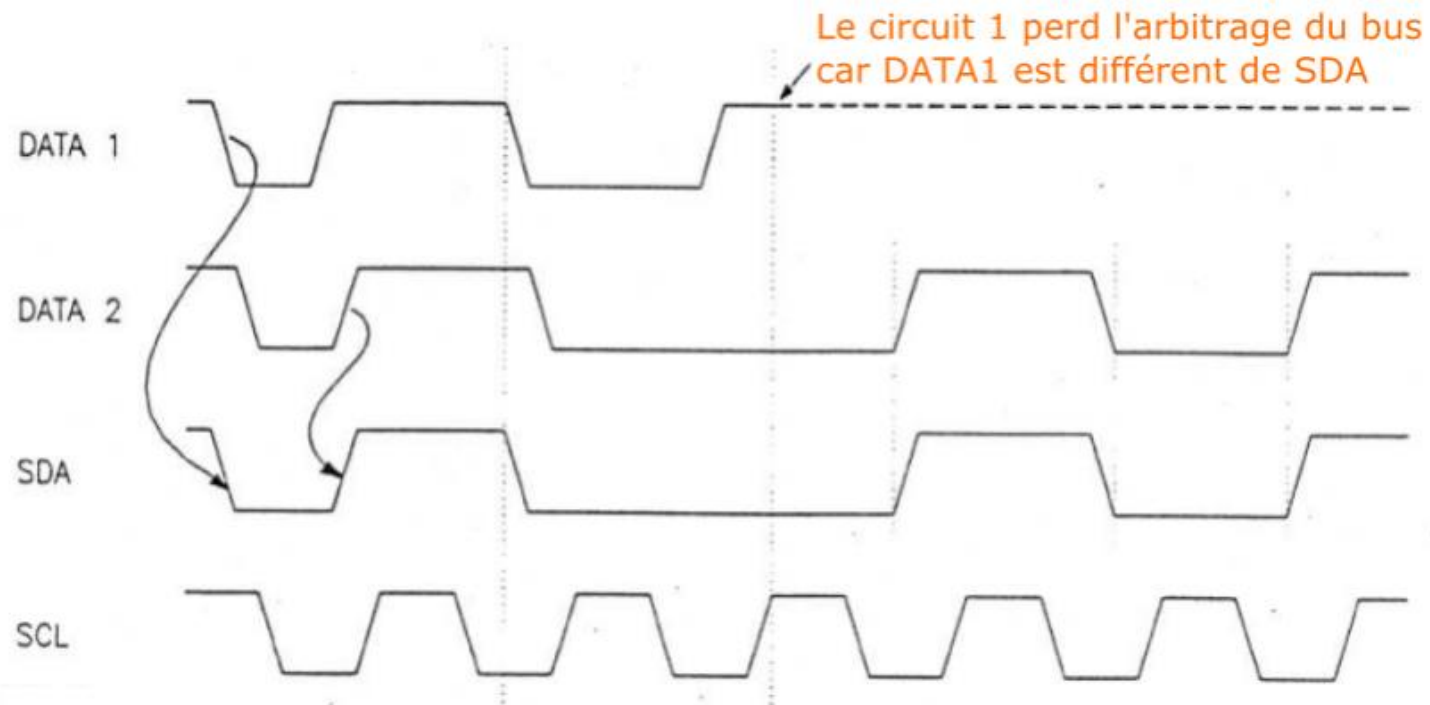
Bus : I2C - Communication

- Cette manière d'imposer une valeur sur le bus implique une dominance/arbitrage de l'information et une hiérarchie des puces.
- Imaginons que deux maîtres veuillent lancer une communication en même temps.
- Ils débutent tous les deux en envoyant une clock sur SCL et transmettent l'information sur le bus.
- Le maître qui l'emporte est celui qui retrouve sur SDA l'information qu'il souhaite mettre le bus. Par cette rétroaction, celui qui perd arrête d'essayer de transmettre.

Comment savoir celui qui "gagne" ?

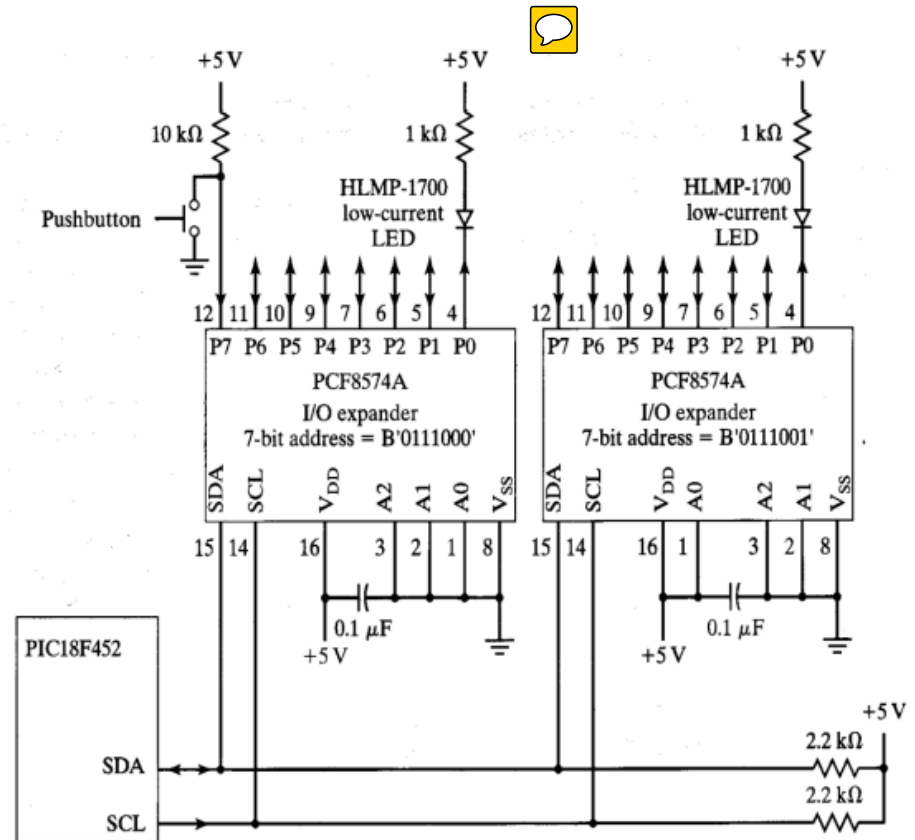
Bus : I2C - Communication

- Celui qui gagne est le premier qui impose un "0" alors que l'autre veut imposer un "1". En effet, le NMOS est dominant par rapport à la résistance de pull-up!



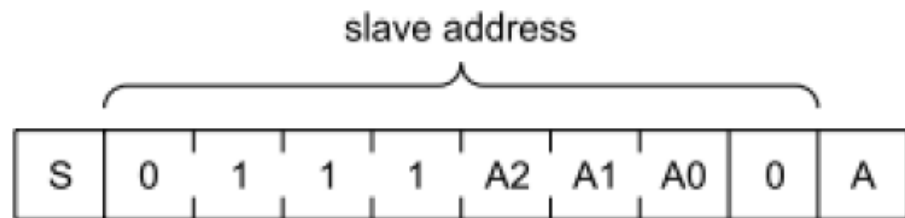
Bus : I2C - Adressage

- Une puce (maître ou esclave) peut écrire ou lire l'information sur SDA.
- Le protocole de communication est tel que c'est toujours le maître qui commence la communication. Il transmet sur SDA, l'adresse de l'esclave qu'il cherche à atteindre.
- **Comment fixer une adresse alors que plusieurs puces du même type peuvent être présentes ?**



Bus : I2C - Adressage

- En I2C, l'adresse d'une puce est composée de 7 bits :
 - Les 4 bits les plus significatifs sont donnés par le constructeur.
 - Les 3 bits les moins significatifs sont fixés lors de la conception du PCB. Ce choix permet de mettre une dominance dans les puces!
 - Pour certaine puce l'adresse peut être sur 10 bits.

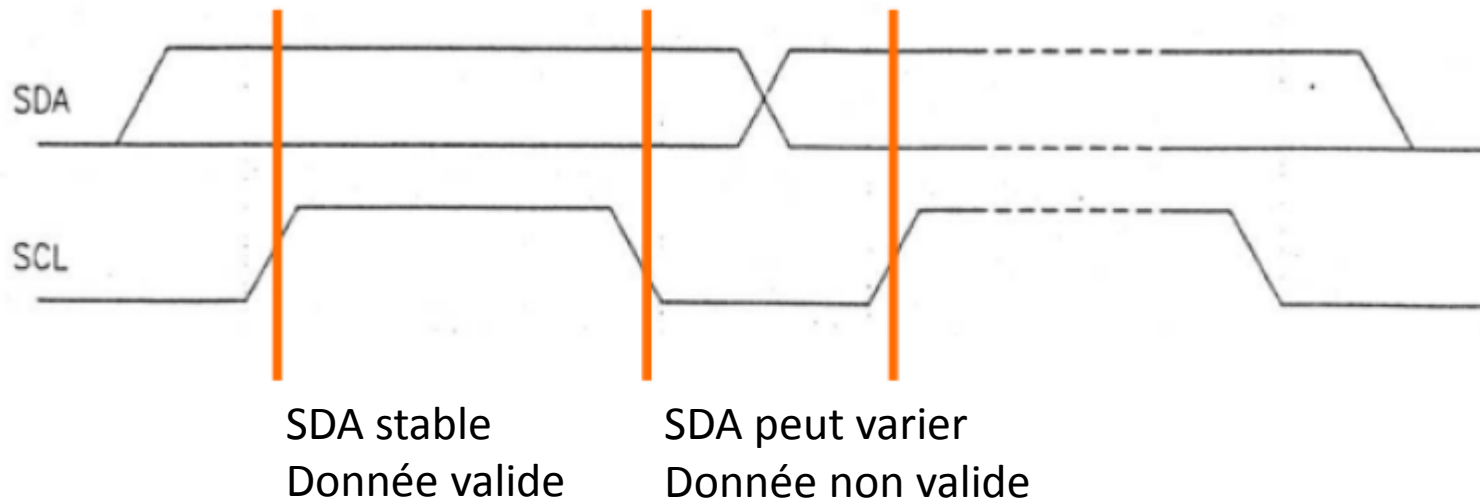


MBD973

b. PCF8574A.

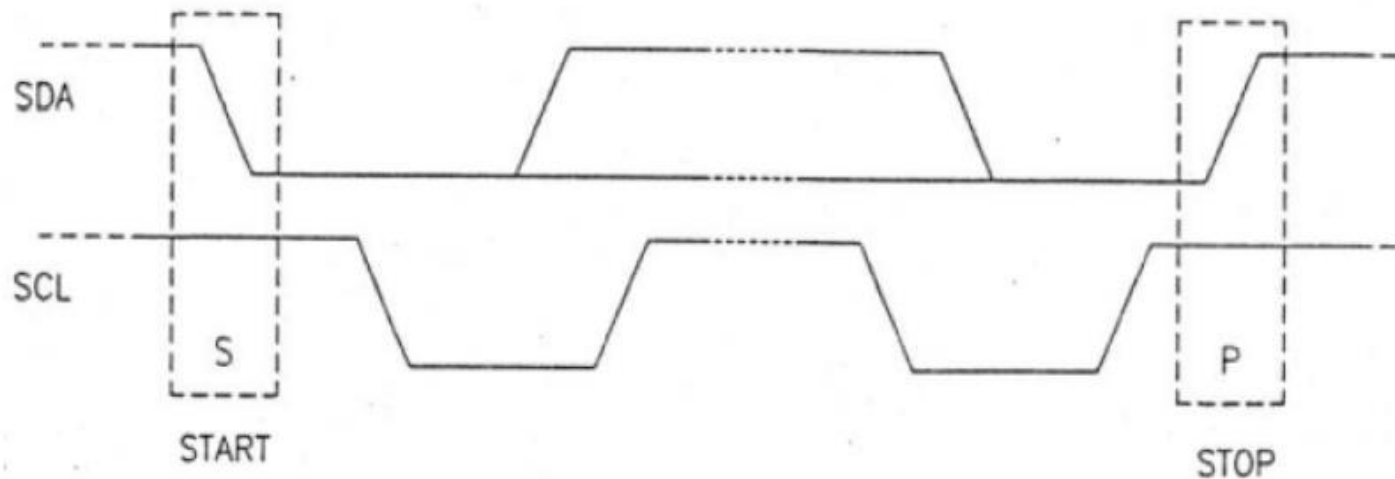
Bus : I2C - Transmission

- L'information sur le bus est échantillonné par le récepteur, lorsque le signal de SCL est haut. L'émetteur assure que l'information est valide.
- Les transitions ne peuvent donc se faire uniquement lorsque la clock SCL est basse



Bus : I2C - Transmission

- Le seul moment de la transmission qui autorise une transition est lors du démarrage et de la fin de la communication (trame)
 - Start bit = flanc descendant de SDA quand SCL est haut
 - Stop bit = flanc montant de SDA quand SCL est haut




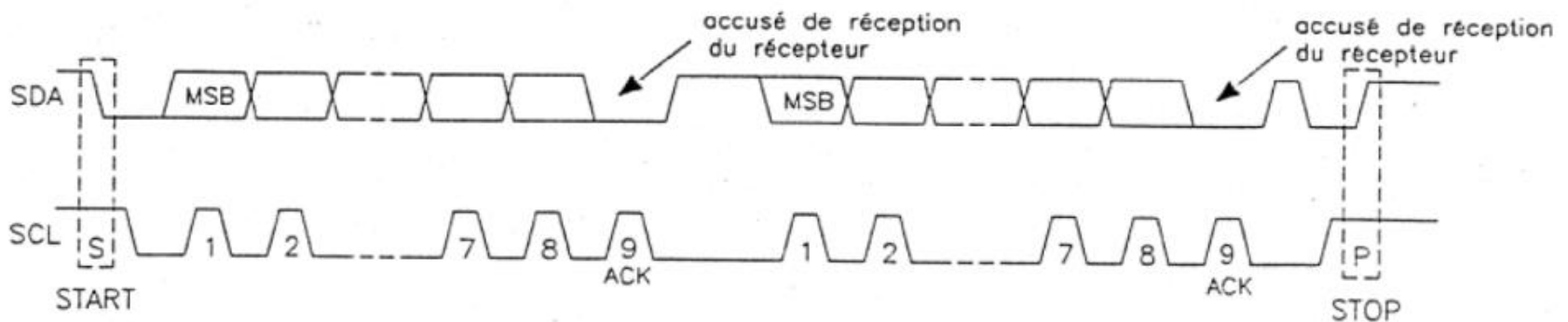
Bus : I2C – Mise en pause

À tout moment, l'esclave peut demander une pause, pour arrêter la transmission du maître. Il empêche le maître de passer la clock à l'état haut en la maintenant à l'état bas. La communication reprend dès que l'esclave relâche l'état bas.



Bus : I2C - Trame

- Le protocole d'une trame est le suivant :
 - Le maître initialise la communication par un Start bit.
 - Des mots de 8 bits sont transmis par l'émetteur (maître et/ou esclave) en partant du MSB.
 - Entre chaque mot, pour  accuser sa bonne réception, le récepteur transmet un "acknowledge" (mise à 0 de SDA)
 - Le maître envoie le stop bit



Bus : I2C – Ecriture

- La séquence pour **écrire** dans une puce esclave :
 - 1. Envoie de l'adresse de la puce + '0'. Ce '0' spécifie une écriture.
 - 2. Envoie de l'adresse N du registre dans la puce.
 - 3. Envoie des informations à écrire. Le premier mot de 8 bits est sauvegardé à l'adresse N, le suivant à l'adresse N+1,...
 - A chaque étape, le récepteur répond à l'émetteur via un acknowledge.

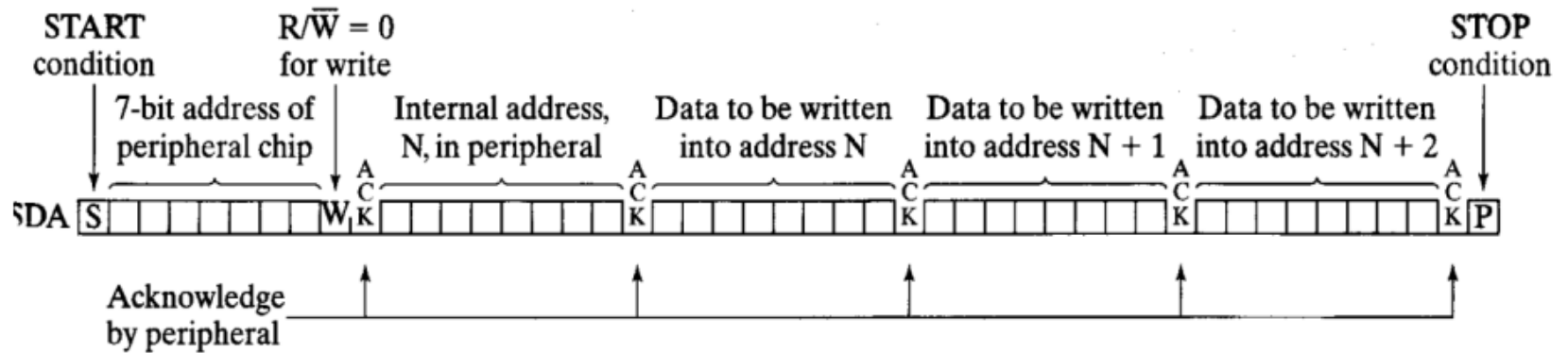
Bus : I2C – Lecture

- La séquence pour **lire** dans une puce esclave :
 - 1. Envoie de l'adresse de la puce + '0'. Ce '0' spécifie une écriture.
 - 2. Envoie de l'adresse N du registre dans la puce. La transmission s'achève pour laisser le temps de charger la donnée désirée.
 - 3. La transmission recommence avec un start bit, le maître envoie l'adresse de la puce + '1'. Ce '1' spécifie une lecture.
 - Aux étapes 1, 2 et 3, l'esclave répond au maître via un acknowledge.

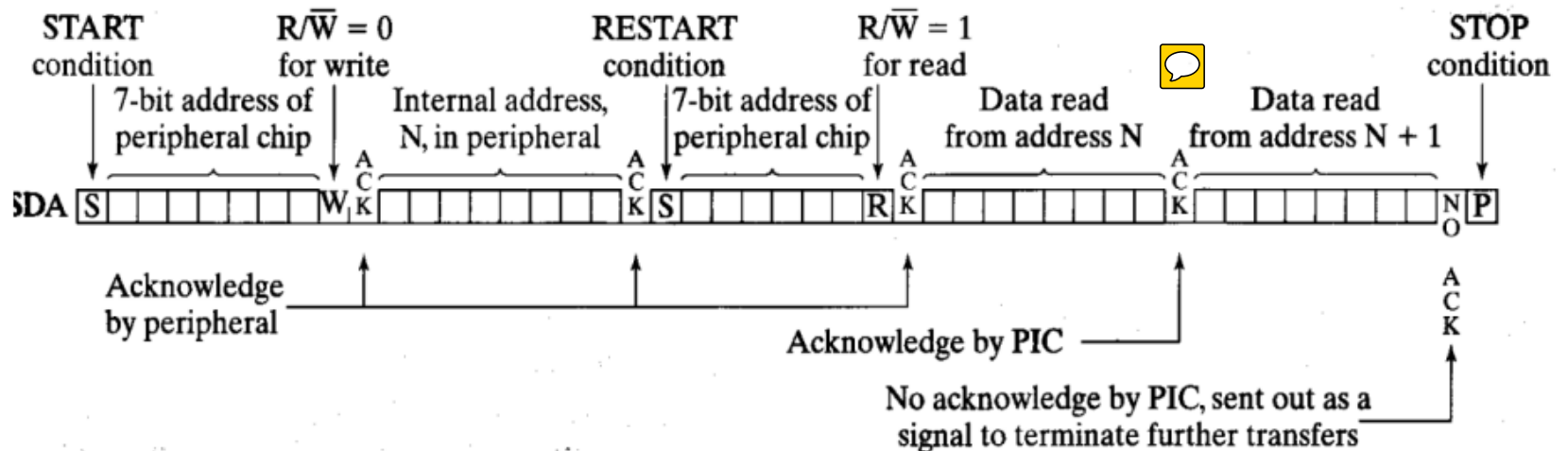


Bus : I2C – Lecture

- La séquence pour **lire** dans une puce esclave :
 - 4. L'esclave transmet les mots au maître. Le premier mot de 8 bits transmis est celui de l'adresse N, le suivant à l'adresse N+1,...
 - 5. Après chaque mot, la maître envoie un acknowledge, sauf après le dernier mot qu'il voulait lire. Ce No Acknowledge, donne l'information à l'esclave de s'arrêter.
 - 6. Un stop bit est envoyé par le maître.



(a) General format to write to several peripheral internal registers or addresses



(b) General format to read from several peripheral internal registers or addresses

Adresses réservées

| 10 bit addresses, binary noted, MSB is left | Purpose |
|---|------------------------------------|
| 0000000 0 | General Call |
| 0000000 1 | Start Byte |
| 0000001 X | CBUS Addresses |
| 0000010 X | Reserved for Different Bus Formats |
| 0000011 X | Reserved for future purposes |
| 00001XX X | High-Speed Master Code |
| 11110XX X | 10-bit Slave Addressing |
| 11111XX X | Reserved for future purposes |