Protocols de Communication Série : I²C, SPI, UART CAN, LIN, FlexRay.

Par: Mehdi Bahlaoui

I²C

I²C (Inter-Integrated Circuit), développé par Philips en 1982 est un protocole série **synchrone (clock)** de type **(multi master-multi-slave),** utilisé principalement pour les communications de données sur de courtes distances.

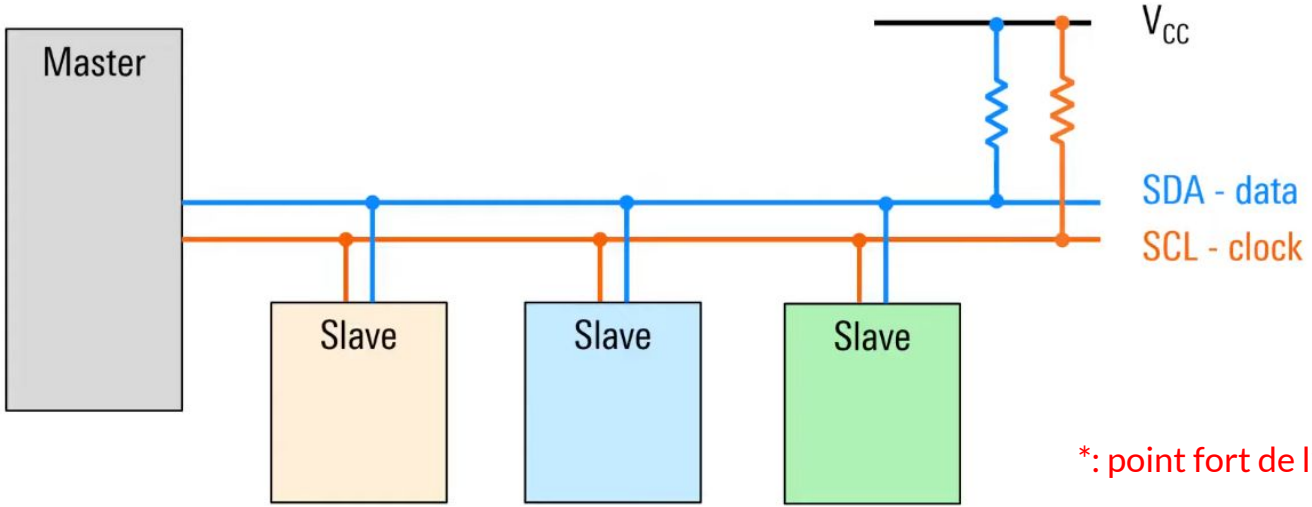
Il fonctionne en **half-duplex**
(les deux dispositifs peuvent transmettre, mais pas simultanément).

Utilisant seulement deux fils :
le Serial Data (SDA) et le Serial Clock (SCL).

Exemple: Le module MPU-6050 utilisant le protocole I²C pour communiquer avec un microcontrôleur.

Topologie

Un maître est connecté à un ou plusieurs esclaves* via deux lignes partagées, SDA et SCL, chacune connectée à une tension VCC par une résistance de pull-up. Les appareils peuvent être ajoutés ou retirés du bus à tout moment.



*: point fort de l'I²C

Fonctionnement

1.) Condition de départ : Un maître revendique le bus en tirant d'abord SDA puis SCL vers le bas.

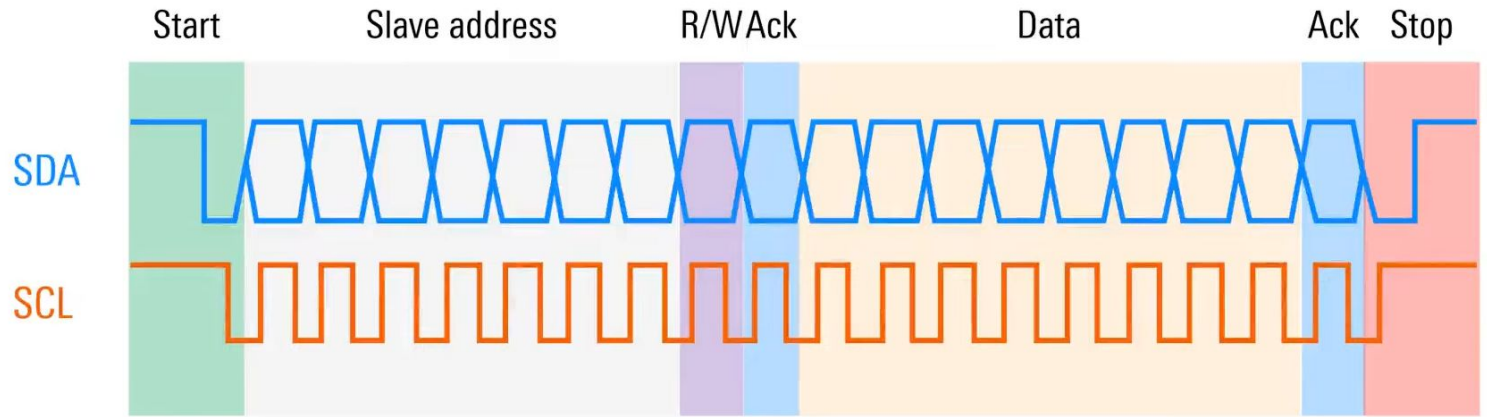
2.) Adresse de l'esclave : Le maître envoie l'adresse de l'esclave avec lequel il souhaite communiquer.
Bit de lecture/écriture : Indique si le maître veut lire ou écrire des données.

3.) Bit de lecture/écriture (R/W) : Indique si le maître veut lire ou écrire des données.

4.) Acknowledge (ACK) : Envoyé par le récepteur après chaque octet de données reçu pour confirmer la réception correcte.

5.) Données : Les données réelles sont transmises sous forme d'octets de 8 bits.

6.) Condition d'arrêt : Indique la fin de la communication en permettant à SCL de revenir à l'état haut puis à SDA.



Modes de vitesse

Outre que le mode utilisé, les valeurs des résistances de pull-up influencent la vitesse maximale du bus et la consommation d'énergie.

Généralement I²C peut fonctionner à différentes vitesses de bus, de Standard Mode (100 kbps) à Ultra-Fast Mode (5 Mbps).

Les dispositifs compatibles avec un mode peuvent souvent fonctionner à des vitesses inférieures.

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra Fast Mode	5 Mbps

SPI

SPI (Serial Peripheral Interface), développé par Motorola dans les années 1980 est une **interface série à quatre fils.**

SPI offre des améliorations de vitesse par rapport à d'autres protocoles de données série comme UART ou I^2C et peut également prendre en charge la **communication full-duplex (Les deux dispositifs peuvent transmettre simultanément)**.

SPI est de type **(One Master - Multi Slave)**

Exemple: Le module RC522 (RFID) utilisant le protocole SPI pour communiquer avec un microcontrôleur.

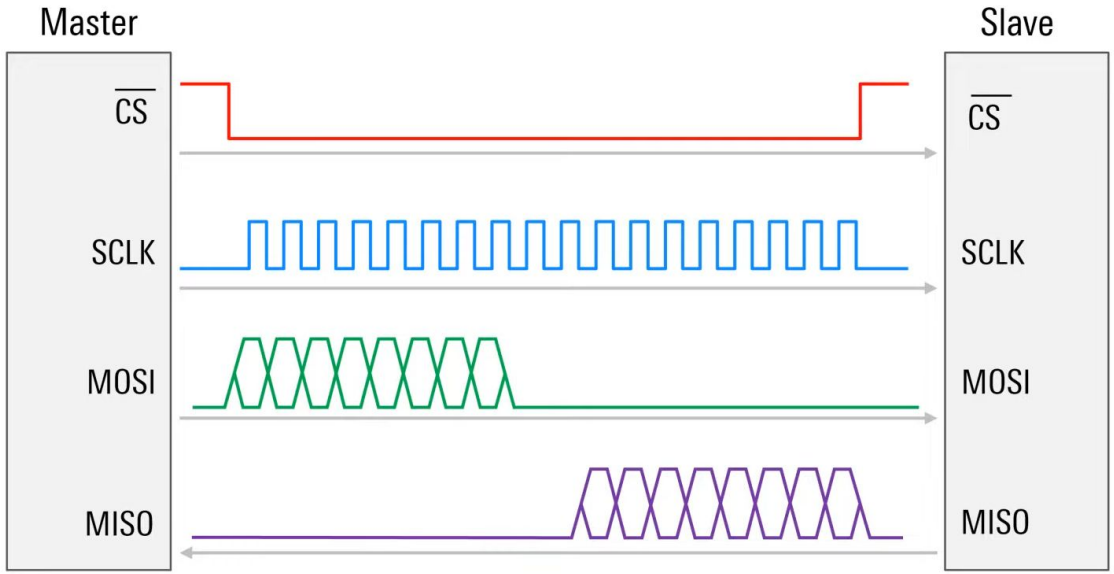
Topologie et Fonctionnement

Ligne Chip Select (CS) : Le maître tire la ligne CS vers le bas pour indiquer à l'esclave que des données sont en cours de transmission, puis démarre le signal d'horloge.

Synchronous Clock (SCLK) : Le maître génère le signal d'horloge, qui peut être actif haut ou bas, indiquant à quel moment l'esclave doit échantillonner les données.

Transmission des données :

- **MOSI** : Utilisé pour envoyer des données du maître à l'esclave.
- **MISO** : Utilisé pour envoyer des données de l'esclave au maître en réponse aux données reçues sur MOSI.



Mode SPI

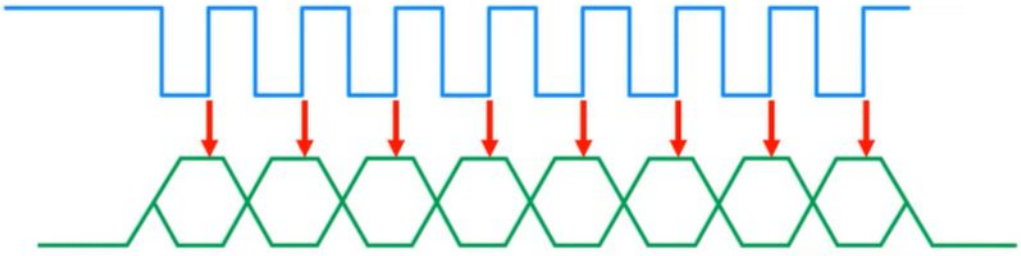
Les combinaisons de **CPOL** et **CPHA** donnent quatre modes possibles (0 à 3), devant être identiques pour tous les dispositifs SPI connectés.

- **Polarité de l'horloge (CPOL) :** Indique si l'horloge est active haute (CPOL=1) ou active basse (CPOL=0).
- **Phase de l'horloge (CPHA) :** Indique si les données sont échantillonnées sur le front montant ou descendant du signal d'horloge.

CPOL = 1 (clock idle high)

CPHA = 1 (read on trailing edge)

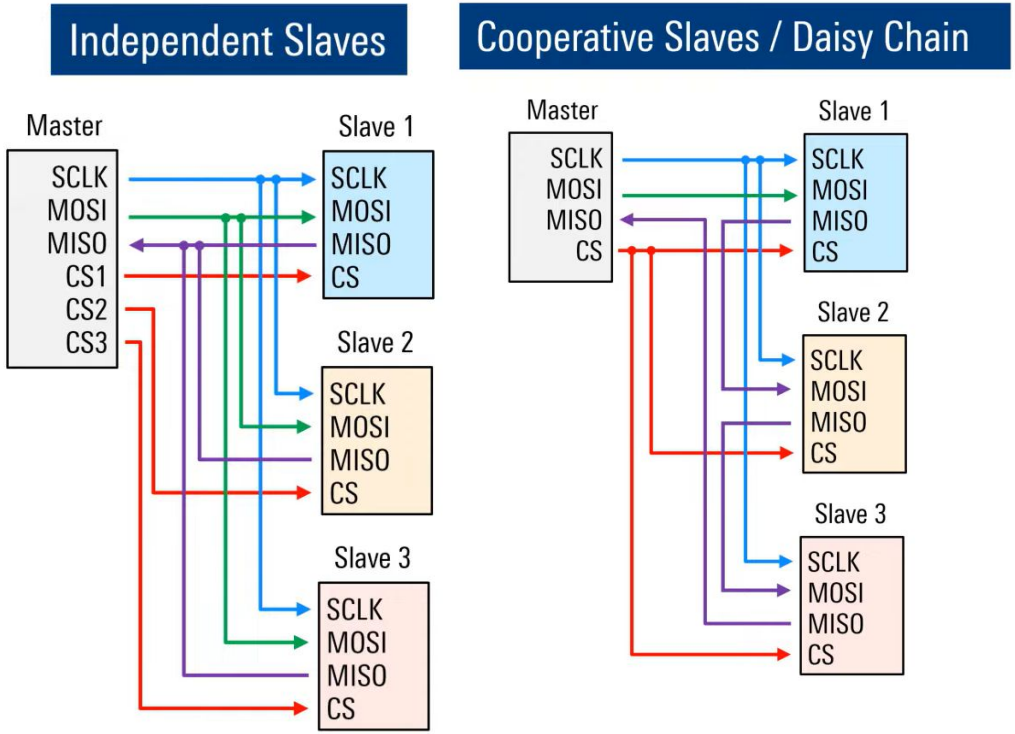
Therefore, mode = 3



Deux Configurations multi-esclaves

- **Esclaves indépendants** : Chaque esclave a sa propre ligne CS.

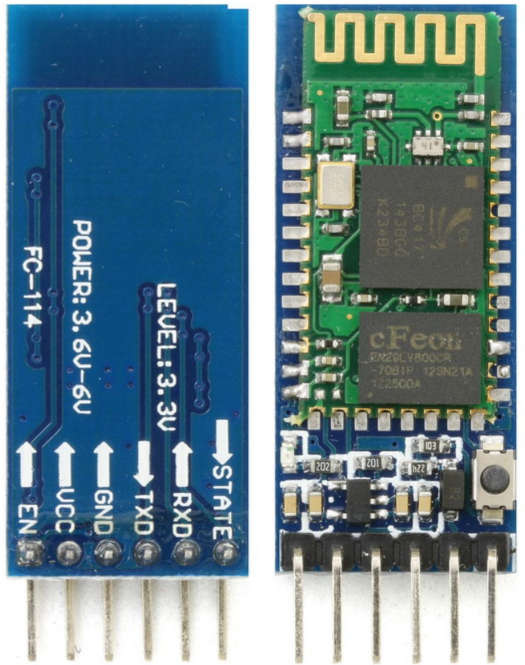
- **Esclaves coopératifs (Daisy Chain)** : Les esclaves partagent une seule ligne CS, avec MOSI du maître connecté au premier esclave, et MISO des esclaves connectés en série jusqu'au maître.



UART

UART (Universal Asynchronous Receiver-Transmitter) est un protocole série **asynchrone** utilisant **deux fils** pour transmettre des données série entre deux dispositifs, un fil pour la transmission (TX) et un pour la réception (RX).

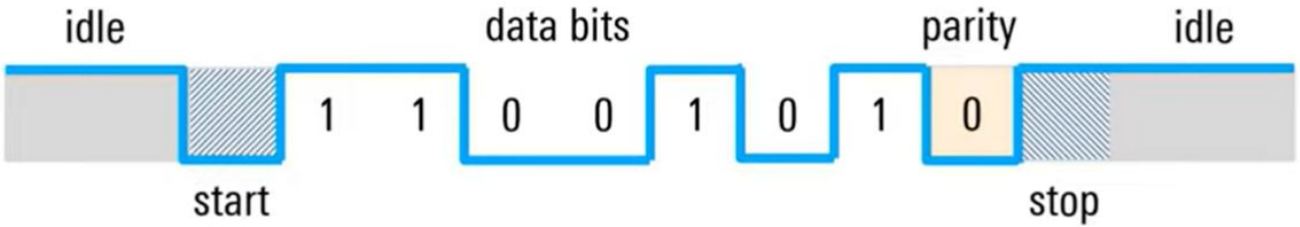
Bien que l'utilisation de UART ait diminué avec l'avènement des protocoles comme SPI, I^2C, Ethernet et USB, il reste pertinent pour des applications à basse vitesse et faible débit en raison de sa simplicité et de son faible coût.



Exemple: Le module HC-05 utilisant le protocole UART pour communiquer avec un microcontrôleur.

Fonctionnement

1. **État de repos** : La ligne est maintenue à un état haut lorsqu'aucune donnée n'est transmise.
2. **Bit de démarrage** : Transition de haut à bas signalant l'arrivée de données.
3. **Bits de données** : Transmis avec le bit le moins significatif en premier.
   Ils contiennent les données utilisateur, généralement entre 5 et 9 bits, souvent 7 ou 8 bits.
4. **Bit de parité (optionnel)*** : Assure que le nombre total des ‘1’ dans la trame est pair (parité paire) ou impair (parité impaire).
5. **Bit d'arrêt** : Retour à l'état de repos haut, signalant la fin de la trame.



Type, Synchronisation et Baud Rate

UART est **full-duplex** est de type **(point-to-point)**

UART est asynchrone, ce qui signifie que le transmetteur et le récepteur n'utilisent pas un signal d'horloge commun.
Les deux extrémités doivent être configurées à la même vitesse de transmission (baud rate) pour synchroniser la temporisation des bits.

Common UART baud rates
4800
9600
19200
57600
115200

Bit de parité

Pendant la transmission, les données peuvent être altérées par des interférences électromagnétiques ou d'autres facteurs.
Le bit de parité aide à détecter les erreurs :

1. **Transmission** : Le transmetteur envoie les données avec le bit de parité correct.
2. **Réception** : Le récepteur reçoit la trame et compte le nombre de bits ‘1’, y compris le bit de parité.
    - **Parité paire** : Si le nombre total des ‘1’ n'est pas pair, il y a eu une erreur.
    - **Parité impaire** : Si le nombre total des ‘1’ n'est pas impair, il y a eu une erreur.

Exemple de détection d'erreur

Considérons la trame suivante envoyée avec la parité paire : 010110100.

- **Trame reçue** : 01011**1**100 (un bit a changé)
- **Nombre des ‘1’** : 5 (impair, donc une erreur est détectée)

Résumé

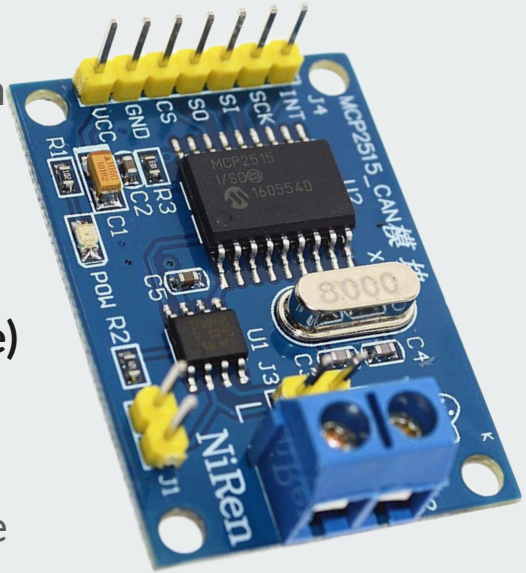
	UART	I2C	SPI
Number of Pins	2	2	>3
Baud Rate (b/s)	Up to 115,200	Up to 400,000	Up to a few megabits
Communication type	Point to Point	Multi Master - Multi Slave	One Master - Multi Slave
Half and Full duplex	Full duplex	Half Duplex	Full Duplex
Synchronous or Asynchronous	Asynchronous	Synchronous	Synchronous
Maximum Number of Devices on the Bus	2	Up to 128	Theoretically Infinite (Limited by the Number of I/O Pins of the Master)
Complexity	Low	High	Medium
Cost	Low	Medium	High

CAN

- **CAN** (Controller Area Network) est à la fois une norme et un protocol de **communication série** utilisée principalement dans les systèmes embarqués, notamment les véhicules automobiles.

  Il utilise une **topologie en bus** (**Multi Master - Multi Slave**) où tous les nœuds partagent le même canal de communication.

  CAN est **half duplex**, et **asynchrone**, permettant à chaque nœud de transmettre des données indépendamment.
- **CAN FD (Flexible Data Rate)** permet des taux de transmission de données plus rapides et des tailles de trame plus grandes que le CAN traditionnel.

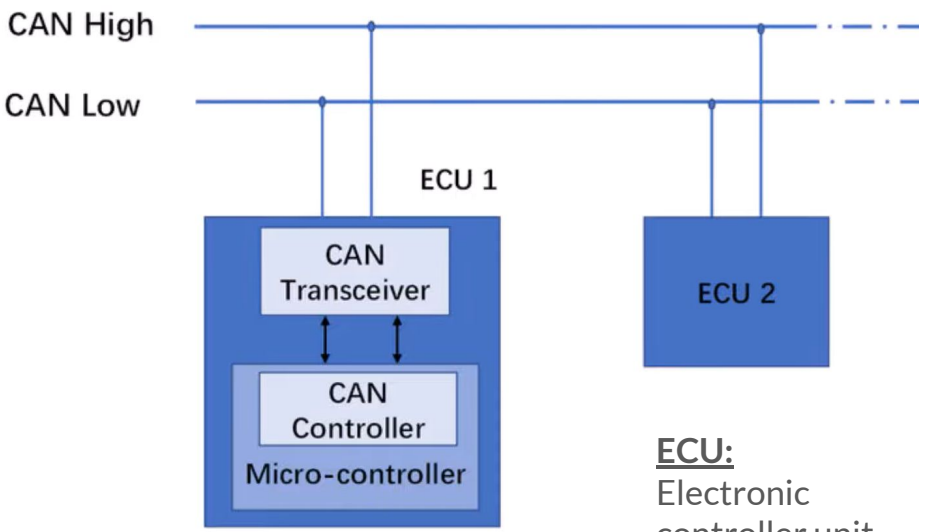


Exemple: Le module MCP2515 servant comme interface SPI-CAN pour utilisation avec un microcontrôleur.

Topologie

Chaque nœud est connecté à un bus commun et peut envoyer et recevoir des messages.
Lorsqu'un nœud veut envoyer un message, il vérifie d'abord si le bus est libre.

Si plusieurs nœuds transmettent en même temps, **un protocole d'arbitrage** permet au message avec la plus haute priorité de continuer **(non destructive method)**.
Tous les nœuds reçoivent le message, mais seuls ceux pour lesquels il est pertinent le traitent.

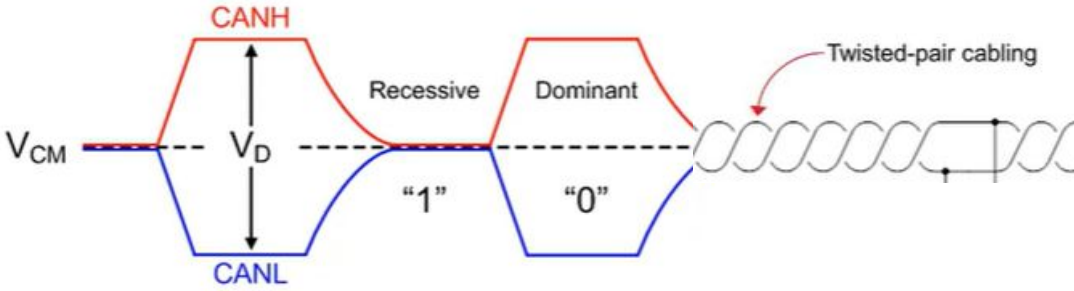


ECU:
Electronic controller unit

Ce système garantit une communication fiable et sans collision, avec une gestion efficace des priorités.

Transmission: Communication différentielle!

CAN utilise **deux fils**, CANH (high) et CANL (low), pour **transmettre des données sous forme de différence de tension** entre eux, plutôt que par rapport à une référence commune.



- Lorsque le bus est inactif, les tensions sur CANH et CANL sont égales.
- Pour envoyer un bit récessif (logique 1), les tensions sur CANH et CANL sont proches l'une de l'autre.
- Pour un bit dominant (logique 0), CANH est tiré à une tension plus élevée tandis que CANL est tiré à une tension plus basse, créant une différence de tension significative entre les deux fils.

Cette méthode de transmission, **combinée à l'utilisation de câbles torsadés**, rend le système CAN extrêmement **résistant aux interférences électromagnétiques**, car toute perturbation affectant les deux fils de manière égale sera annulée.

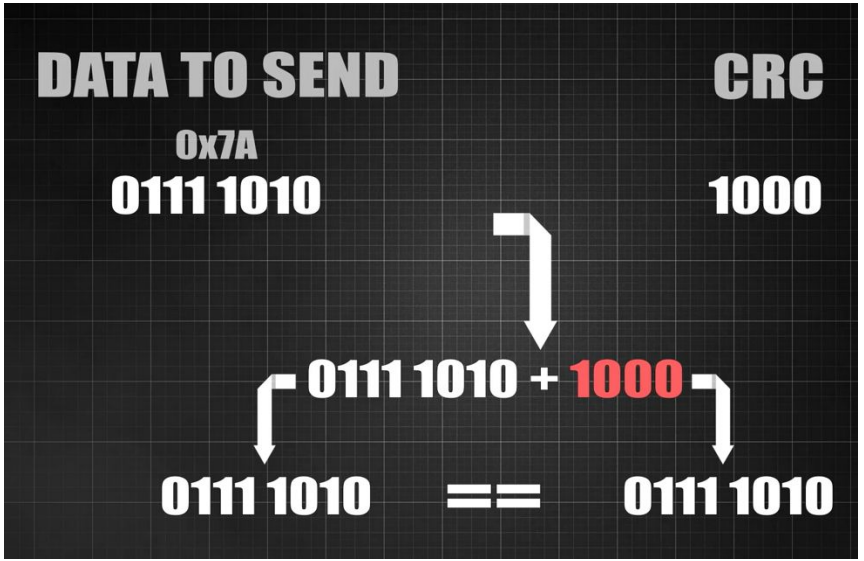
CRC (Contrôle de Redondance Cyclique)

Même si la lecture différentielle échoue, nous pouvons toujours vérifier si les données reçues sont correctes.

La méthode est la suivante:

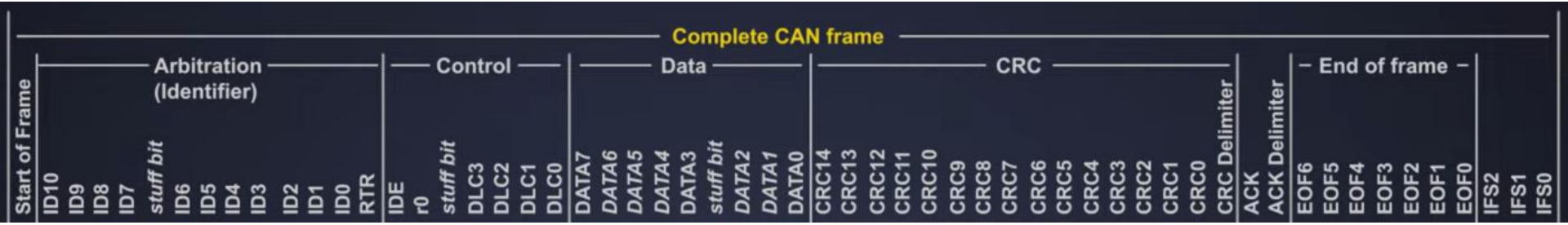
1) On calcul un terme spécial appelé CRC à partir d'une formule et à base des données reçues.

2) Le récepteur obtient les données et le CRC, puis effectue un calcul inverse pour obtenir le CRC des données reçues.
Si les CRCs ne correspondent pas, cela signifie qu'il y a eu une erreur dans la transmission des données.



Cette capacité du CAN à résister aux interférences et à garantir la sécurité des données est une raison majeure de son utilisation dans les automobiles.

Fonctionnement

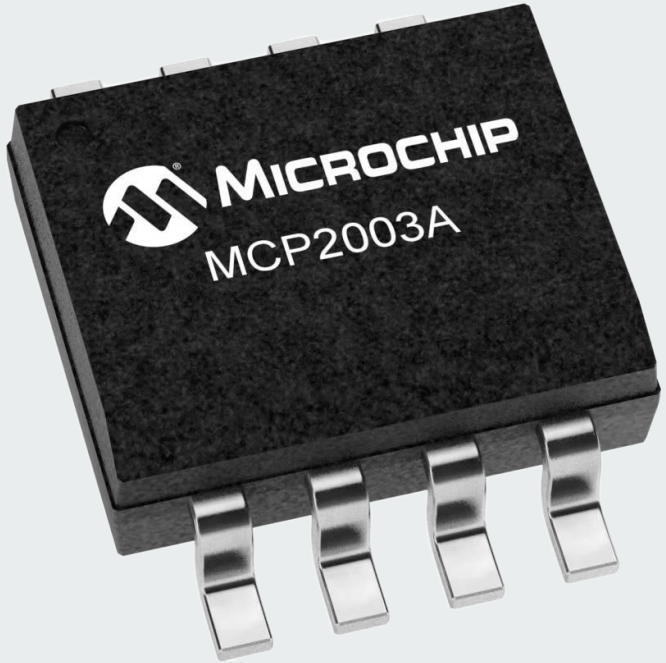


1. **Start of Frame:** Indique le début de la trame.
2. **Arbitration:** Utilisé pour arbitrer les messages sur le bus lorsqu'il y a des tentatives simultanées de transmission par plusieurs nœuds.
Le bit RTR Indique si la trame est une demande de données (récessif) ou une trame de données (dominant).
3. **Control:** -Le bit IDE: indique si on utilise Standard CAN ou Extended CAN.
   -Le bit r0: utilisé pour des mesures futures de compatibilité.
   -Les bits DLC: Permettent d'indiquer la taille du message transmis.
4. **Data:** Transporte les données utiles.
5. **CRC:** Détecte les erreurs de transmission.
6. **ACK:** Confirme la réception correcte des données.
7. **End of Frame:** Indique la fin de la trame.
8. **IFS:** Servent à séparer les trames de données successives sur le bus CAN.

Lin

LIN (Local Interconnect Network) est à la fois une norme et un protocole de communication série **asynchrone**.
Il utilise une **topologie (One Master - Multi Slave)**.

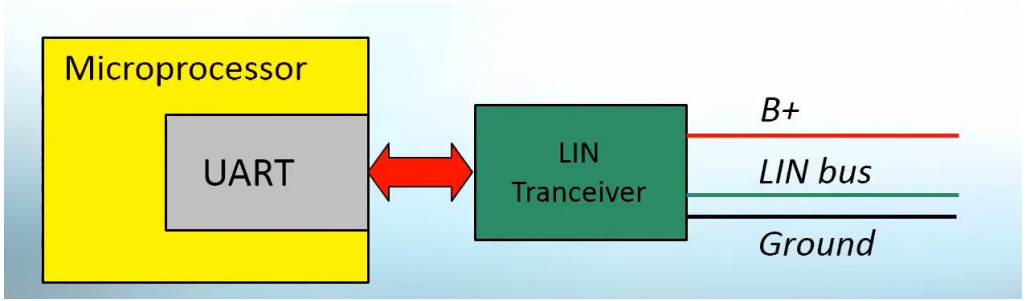
LIN est conçu pour permettre une transmission de données **fiable** et **économique**, avec des **temps de latence garantis** et s**ans besoin d'arbitrage** pour les messages.
Il est souvent utilisé comme **sous-réseau dans un réseau CAN**, permettant aux constructeurs automobiles de transférer des fonctions non critiques, à faible vitesse et non sécuritaires, du réseau CAN à un réseau LIN.



Exemple: Le module MCP2003A servant comme interface UART-LIN pour utilisation avec un microcontrôleur.

Topologie

Les réseaux LIN sont implémentés à l'aide d'interfaces UART, courantes et peu coûteuses dans les microcontrôleurs. Cela rend le réseau facile à mettre en œuvre, nécessitant uniquement un fil pour la communication et un transceiver LIN pour gérer les niveaux de tension.
(Contrairement au CAN, qui requiert un contrôleur CAN pour la communication différentielle et un transceiver CAN)



Deux Configurations

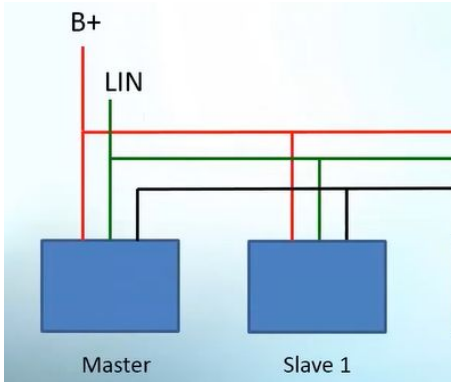
Un réseau LIN typique comprend un maître et jusqu'à **15 esclaves**, bien que l'ajout d'un plus grand nombre d'esclaves puisse entraîner des problèmes de chute de tension et d'impédance.

Le maître utilise une résistance de tirage de 1kΩ, tandis que les esclaves utilisent des résistances de 30kΩ.

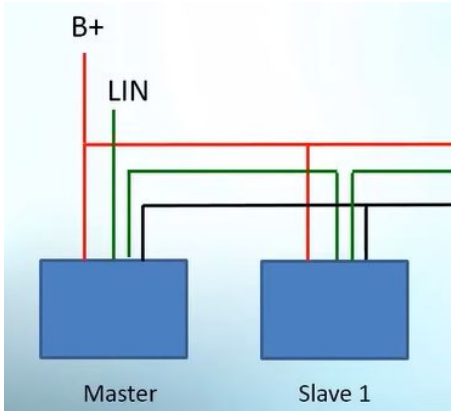
Dans une configuration traditionnelle d'un réseau LIN, **chaque dispositif esclave** doit être manuellement configuré avec une adresse unique.

Cette tâche, bien que essentielle, peut être fastidieuse et sujette à des erreurs, surtout lorsqu'il s'agit de gérer un grand nombre de dispositifs similaires dans un système complexe, comme un véhicule.

En revanche, l'auto-adressage simplifie grandement ce processus. Grâce à cette fonctionnalité, les dispositifs esclaves peuvent déterminer automatiquement leur adresse unique lors de l'initialisation du réseau.



Configuration Normale



Auto-Addressing

Trois Types de Trames

Les trames inconditionnelles sont envoyées à intervalles réguliers et prédéfinis, indépendamment de l'état du système ou des événements qui se produisent.
Elles sont utilisées pour transmettre des informations critiques et récurrentes, assurant ainsi la communication constante de certaines données.

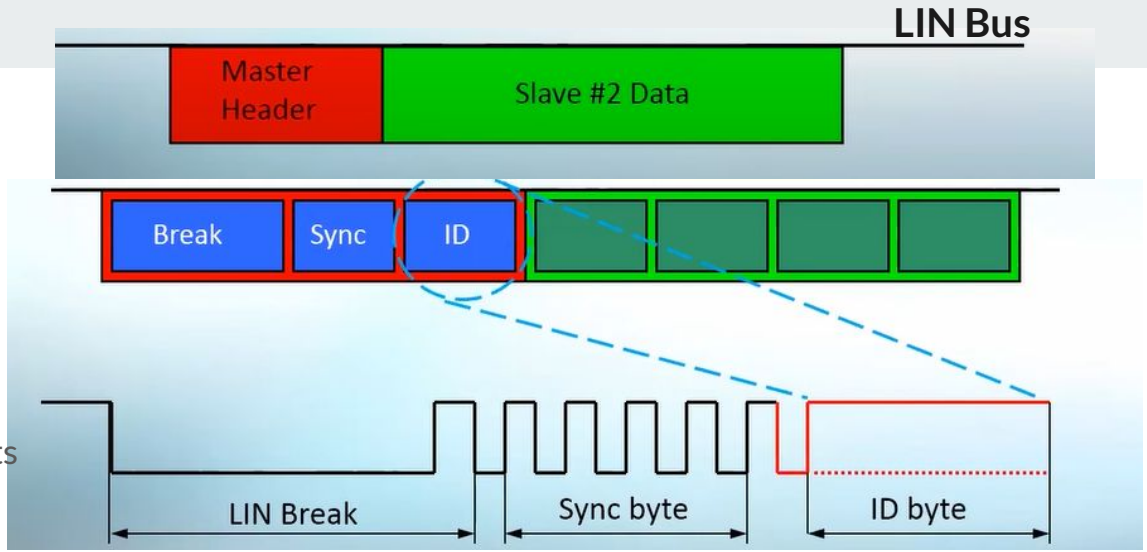
Les trames déclenchées par événement sont émises en réponse à un changement spécifique ou à un événement particulier dans le réseau.
Elles sont moins fréquentes que les trames inconditionnelles et permettent une utilisation plus efficace de la bande passante en envoyant des données uniquement lorsque cela est nécessaire.

Les trames sporadiques sont similaires aux trames déclenchées par événement, mais elles sont envoyées de manière irrégulière, sans intervalles prédéfinis.
Elles sont généralement utilisées pour des messages moins prioritaires qui ne nécessitent pas une communication constante, mais qui doivent tout de même être envoyés lorsque l'événement correspondant se produit.

Topologie

Le maître envoie toujours le début de chaque message, tandis que l'esclave ne peut remplir que les octets de données.

La trame LIN est se compose des éléments suivants :



1. **Champ de rupture (Break):** Signale le début d'une nouvelle trame pour permettre la synchronisation de tous les nœuds **(13 bits LOW minimum + 1 bit HIGH)**.
2. **Champ de synchronisation (Sync) :** Un motif fixe utilisé pour synchroniser l'horloge de tous les nœuds sur celle du nœud maître.
3. **Champ d'identifiant (Protective Identifier):** Contient l'identifiant de la trame, incluant l'adresse du nœud esclave et **2 bits de parité**.
4. **Champ de données :** Contient les données réelles transmises, pouvant aller jusqu'à **8 octets.**
5. **Champ de somme de contrôle :** Assure l'intégrité des données en vérifiant que les données reçues sont identiques à celles envoyées **(à l'aide d'une formule)**.

LIN Vs.CAN

- LIN est plus **lent** : 20 kbps Vs. 1 Mbps
- LIN n'a qu'un seul (**1**) nœud pouvant démarrer le message de transmission et **pas d'arbitrage**
- Le protocole LIN est géré par des **pilotes logiciels**, CAN est géré par des **contrôleurs matériels**
- LIN dispose de **« tableaux de planification »** dédiés qui contrôles l’utilisation du bus

FLEXRAY

FlexRay est un protocole de communication **haut débit** qui, comme CAN utilise **deux fils** dans une architecture de **bus** à réseau **redondante.**
Cela signifie que le réseau possède généralement **deux canaux de communication indépendants** (canal A et canal B) où pour **chaque, une paire de câbles torsadées.**
L'objectif étant clair: Assurer un **fonctionnement continu en cas de défaillance.**

Il peut fonctionner en mode **synchrone**, où les messages sont transmis à intervalles réguliers, **ou en mode asynchrone**, où les messages sont transmis en fonction de la demande.

Comparé à d'autres protocoles comme le CAN ou le LIN, FlexRay offre une bande passante beaucoup plus élevée, pouvant atteindre jusqu'à 10 Mbps **(10x plus vite que CAN)**, et une **latence faible**, ce qui le rend idéal pour les applications de contrôle critiques telles que les **systèmes de freinage électronique**, la direction assistée électrique, et les systèmes de suspension active.
FlexRay utilise une **communication différentielle** similaire à celle de CAN.

Techniques de multiplexage

- Le temps dans FlexRay est divisé en *slots*.
- Ces slots sont répartis en deux segments :
  - Segment statique (TDMA classique) : Allocation fixe et déterministe.
  - Segment dynamique (FTDMA) : Allocation flexible selon la demande.

Cela permet à FlexRay de combiner les avantages d'une communication déterministe avec une certaine flexibilité pour des transmissions moins critiques.

Time Division Multiple Access (TDMA)

Chaque nœud se voit attribuer un ou plusieurs slots TDMA pour transmettre ses données. Voici comment cela fonctionne:

Chaque nœud sait exactement quand il doit transmettre ses données, car les slots sont fixés et prédéfinis dans le cadre du cycle. Cela garantit une **latence faible et prédictible**, essentielle pour les systèmes critiques.

Étant donné que chaque nœud a un slot (ou plus) dédié, il n'y a **pas de risque de collision**, ce qui signifie que les données peuvent être transmises de manière fiable à chaque cycle.

Flexible Time Division Multiple Access (FTDMA)

Contrairement aux slots fixes du segment statique, les slots dans le segment dynamique ne sont pas prédéterminés. Ils sont attribués de manière flexible, en fonction de la demande de transmission des nœuds.

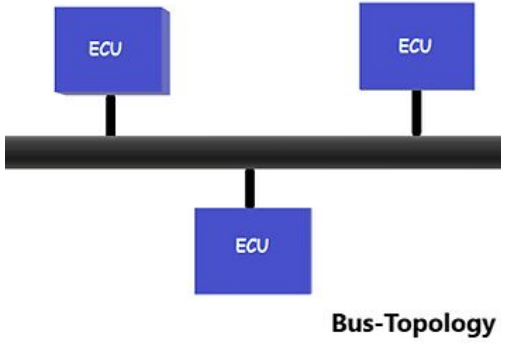
Dans le FTDMA, les nœuds peuvent concourir pour obtenir un slot de transmission.
Chaque nœud se voit attribuer une priorité, et les nœuds ayant la priorité la plus élevée sont servis en premier.
Un algorithme, appelé *Minislots*, est utilisé pour déterminer quel nœud obtiendra le slot disponible.
Si un nœud n'a pas besoin de transmettre, le slot peut être attribué à un autre nœud.

Le FTDMA permet une **utilisation plus efficace de la bande passante**, en attribuant des slots s**elon les besoins actuels de communication**, ce qui est utile pour des systèmes où la charge de travail peut varier.

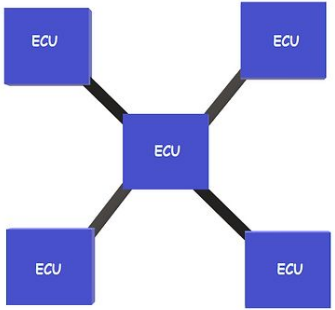
Types de connections

Note: Chaque nœud nécessite un contrôleur flexray et un transceiver Flexray.

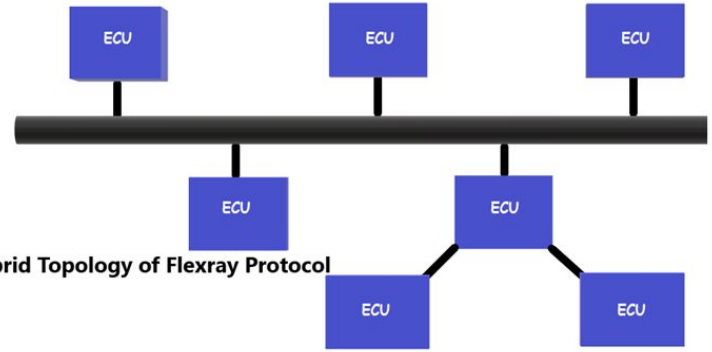
L'un des avantages notables de FlexRay est sa flexibilité d'installation à l'intérieur du véhicule, en fonction de la configuration du véhicule.
Il offre plusieurs options de topologie, telles que les connexions en **Bus (passif multi-drop)**, en **Étoile (active)** ou une combinaison de ces deux topologies appelée topologie **hybride.**
Cette flexibilité permet aux concepteurs d'améliorer les performances, d'augmenter la fiabilité et d'optimiser les coûts pour une conception de système de véhicule spécifique.



Bus-Topology



Star-topology of Flexray Protocol



Hybrid Topology of Flexray Protocol

FLEXRAY Vs. CAN

1)

	CAN	FLEXRAY
	Scalable , Event Driven	Time-Driven , Deterministic, Redundant, Fault -Tolerant , Global Time Base
Medium Access	Multi Master CSMA -CR	Multi Master Hybrid TDMA
Nodes	4-20 Nodes	4-22 Nodes
Bus Speed	33Kbps to 500 Kbps	2.5 - 10 Mbps
Wires	A twisted pair wire	1-2 twisted pair wires
Data Size	0-8 bytes	0-254 bytes
Industry Acceptance	Acceptance in all regions	Limited deployment in NA Wide acceptance in Europe
Cost	Less Cost	More Cost

FLEXRAY Vs. CAN

2)

CAN dispose d'excellents mécanismes de détection et de gestion des erreurs, mais il ne possède pas de dispositif de protection du bus.
En revanche, FlexRay intègre un **(Bus Guardian),** qui joue un rôle crucial dans **la sécurité du réseau**.
Ce gardien de bus surveille le trafic et ne permet au transceiver FlexRay de placer les données sur le bus que si ces **données respectent strictement le calendrier de communication** prédéfini.

Cela garantit que les transmissions se déroulent de manière ordonnée et sans interférences, renforçant ainsi la fiabilité du système.

Webographie

Rohde Schwarz on Youtube:
“Understanding I2C” - https://www.youtube.com/watch?v=CAvawEcxoPU
“Understanding SPI” - https://www.youtube.com/watch?v=0nVNwozXsIc
“Understanding UART” - https://www.youtube.com/watch?v=sTHckUyxwp8

Electronoobs on Youtube:
“PROTOCOLS: UART - I2C - SPI - Serial communications #001”
https://www.youtube.com/watch?v=IyGwvGzrqp8

Can:
https://www.youtube.com/watch?v=JZSCzRT9TTo
https://www.youtube.com/watch?v=QYX_XOjjGOM
https://www.youtube.com/watch?v=YBrU_eZM110
https://www.youtube.com/watch?v=4nk8Lb_K8rs

LIN:
https://www.youtube.com/watch?v=TngJZVb33zc
https://www.youtube.com/watch?v=nWoP3tmzNDM (very good)

Flexray:
https://www.youtube.com/watch?v=f-947qWzkSk&t=1624s
https://www.youtube.com/watch?v=HgZ5Dtm_CIo&t=915s
https://www.youtube.com/watch?v=N8ZHn0hkWOE (good)
https://www.youtube.com/watch?v=cTDTEpogV8E (good)
https://www.influxbigdata.in/flexray