Chapitre 7

Bus de communication

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

INTRODUCTION

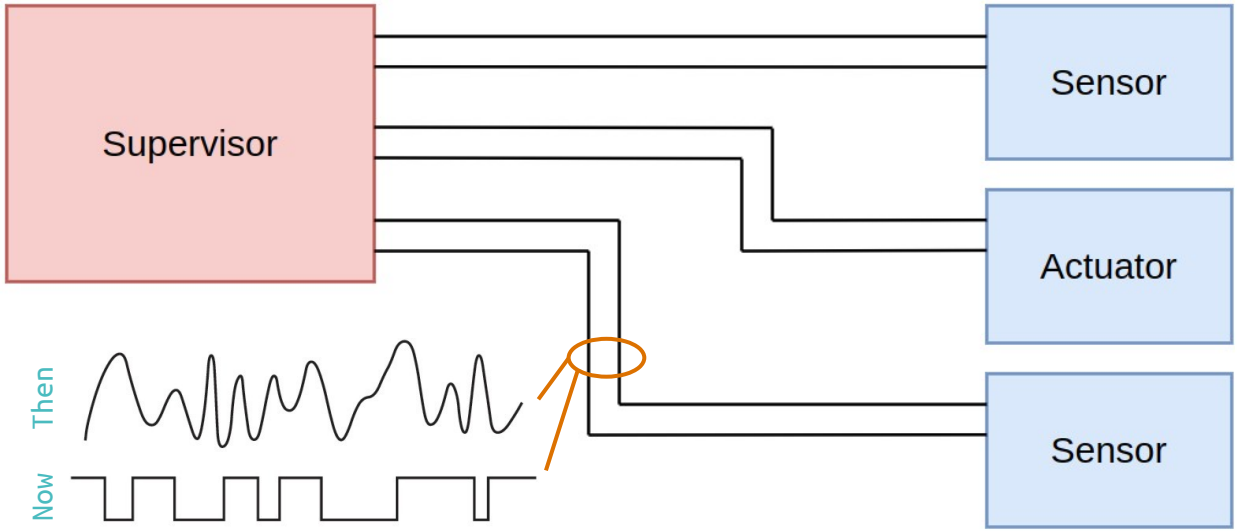
ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Historiquement les informations entre un processeur et un capteur (ou un actionneur) étaient échangées sous forme analogique (par ex, les boucles de courant 4-20 mA).

Toutefois l'information analogique est sensible aux perturbations électro-magnétiques.

En passant sur des **lignes numériques (TOR pour Tout Ou Rien)**, la sensibilité à ces perturbations est grandement réduite tout en obtenant un plus grand débit d'échange.

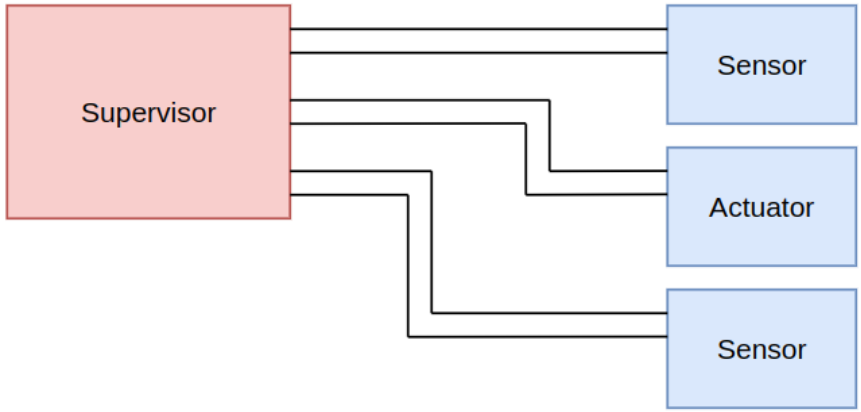


Un autre inconvénient historique est que chaque capteur ou actionneur possédait sa propre liaison avec le processeur. Les systèmes intégrant de plus en plus de périphériques, le nombre de fils aurait été aujourd'hui trop important à intégrer dans un système, tant en terme de coût que de broches disponibles sur le processeur.

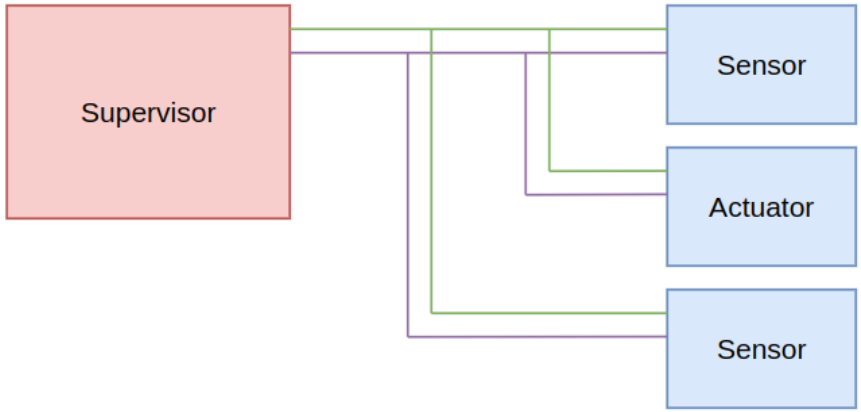
Les bus de communications offrent la possibilité de relier plusieurs capteurs et actionneurs sur les mêmes lignes électriques :

Coût matériel plus faible, temps d'installation plus rapide, meilleure maintenabilité, meilleur évolutivité, …

Separate line for each sensor/actuator



All connected onto the very same line



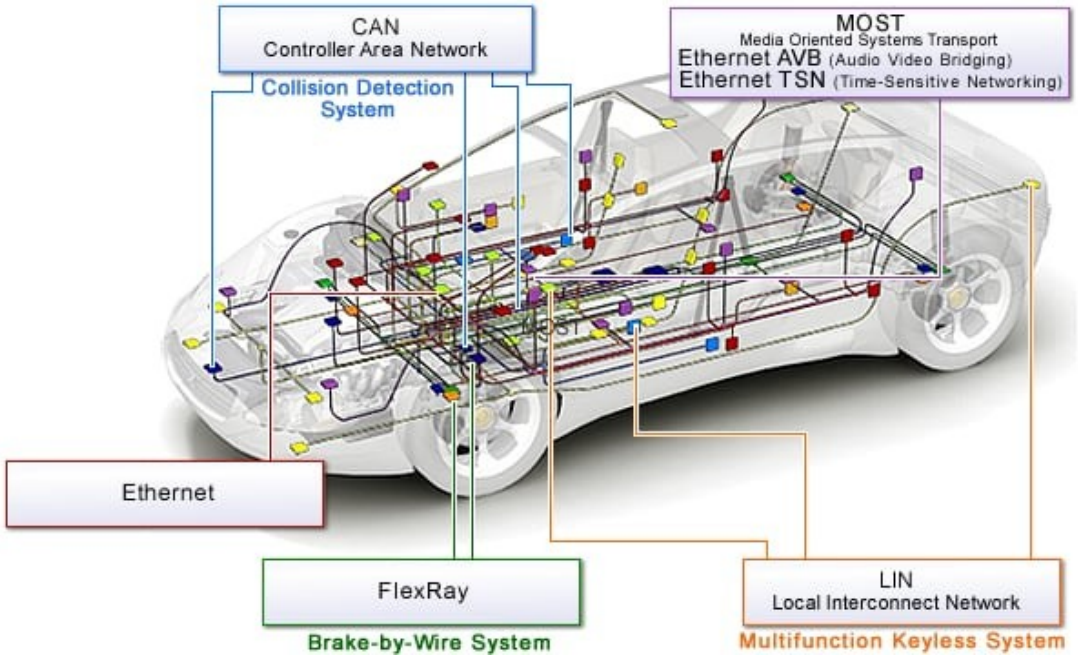
Un bus de communication (en : *bus*) est un **système de communication** entre appareils.

De nombreuses implémentations existent, avec des spécifications matérielles et protocolaires particulières. Leur point commun est que tous les appareils partagent les mêmes fils, rendant le système à la fois économe en coût et facilement évolutif.

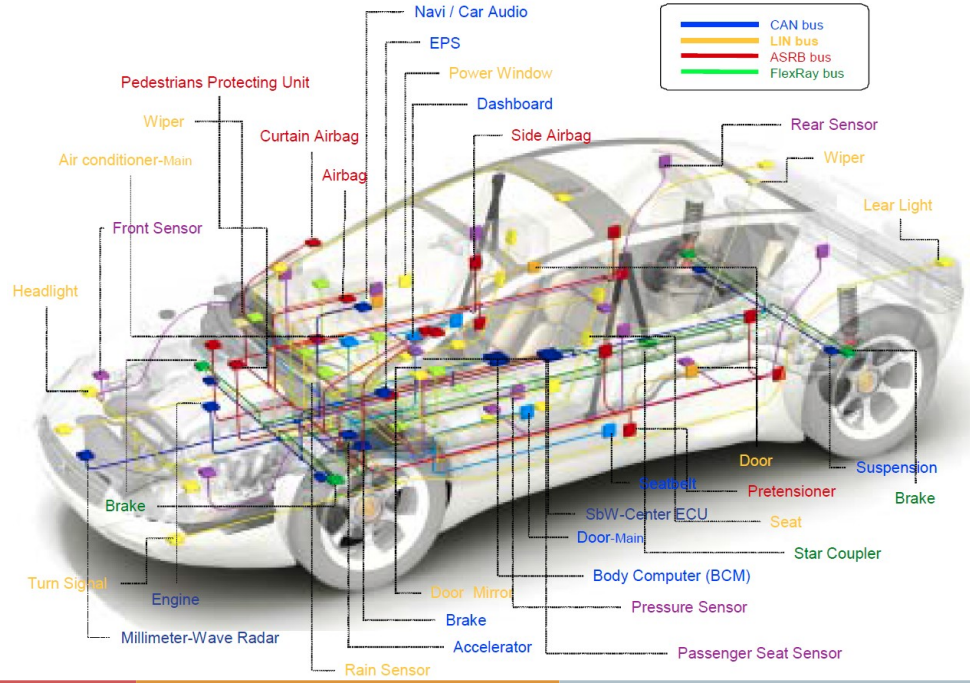
Selon le contexte, le terme bus peut désigner :

- Le matériel (fils, débit, encodage, …) ≈ Couche « Physique » du modèle OSI (7e couche – PHY)
- Le protocole (adressage, contrôle d'erreur, …) ≈ Couche « Liaison » du modèle OSI (6e couche – LINK)
- Le logiciel (driver et/ou stack) ≈ Une des couches hautes du modèle OSI

Exemple d'application : l'automobile



https://www.renesas.com/us/en/application/automotive



https://flexautomotive.net/EMCFLEXBLOG/post/2015/09/08/can-bus-for-controller-area-network

PROPRIÉTÉS DES BUS

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

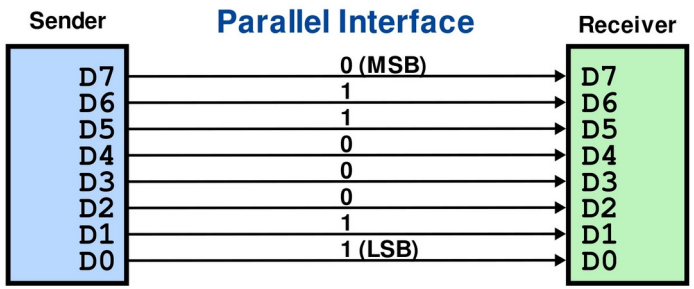
Vocabulaire

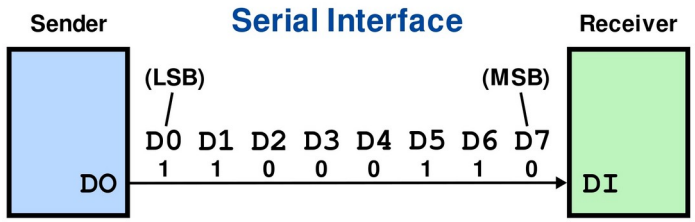
Cette partie du chapitre apporte des définitions sur plusieurs propriétés des bus :

- Communication parallèle vs communication série
- Topologie (point-à-point, bus)
- Mode de transmission (simple, half-duplex, full-duplex)
- Modèle de communication (maître/esclave, client/serveur, producteur/consommateur, publieur/abonné)
- Portée (bus de PCB, bus informatique, bus de terrain)

Communication parallèle vs Communication série

Les bus sont généralement soit **parallèle** soit **série**, même si certaines combinaisons parallèle/série existent (plusieurs liens série en parallèle, ex PCIe, DisplayPort, …).





https://learn.adafruit.com/assets/59130

Afin de simplifier les schémas, toutes les connexions d'un bus sont dessinées sous forme d'un seul trait gras.

Communication parallèle

Les **bus parallèle** sont les plus simples à implémenter d'un point de vue logique :

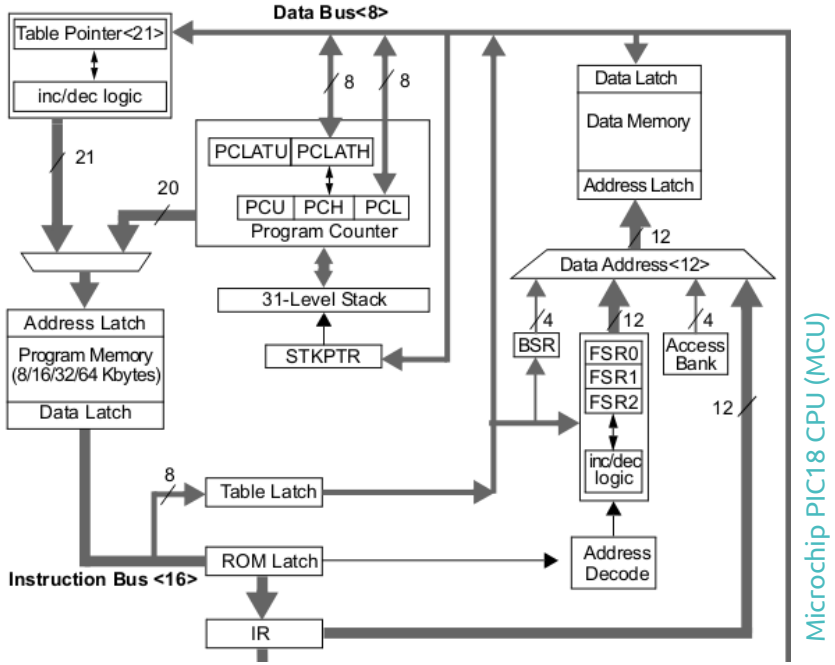
- Chaque entrée/sortie correspond à 1 bit
- Conception matérielle plus simple des périphériques de communication

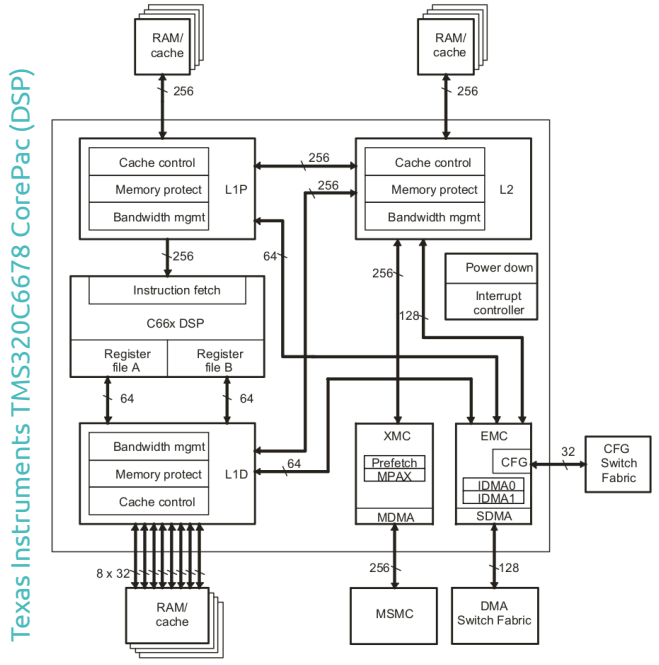
Malgré cela l'utilisation des bus parallèle s'est faite rare pour plusieurs raisons :

- Plus lents qu'un bus série
  - Les interférences entre les lignes compromettent l'intégrité des données sur de longues distances, ou à haute fréquence
- Un grand nombre de broches est nécessaire (pour le processeur et pour le périphérique)
- Grand complexité de routage du PCB
- Grand coût matériel (cuivre sur le PCB, cuivre dans les câbles, connecteurs, …)

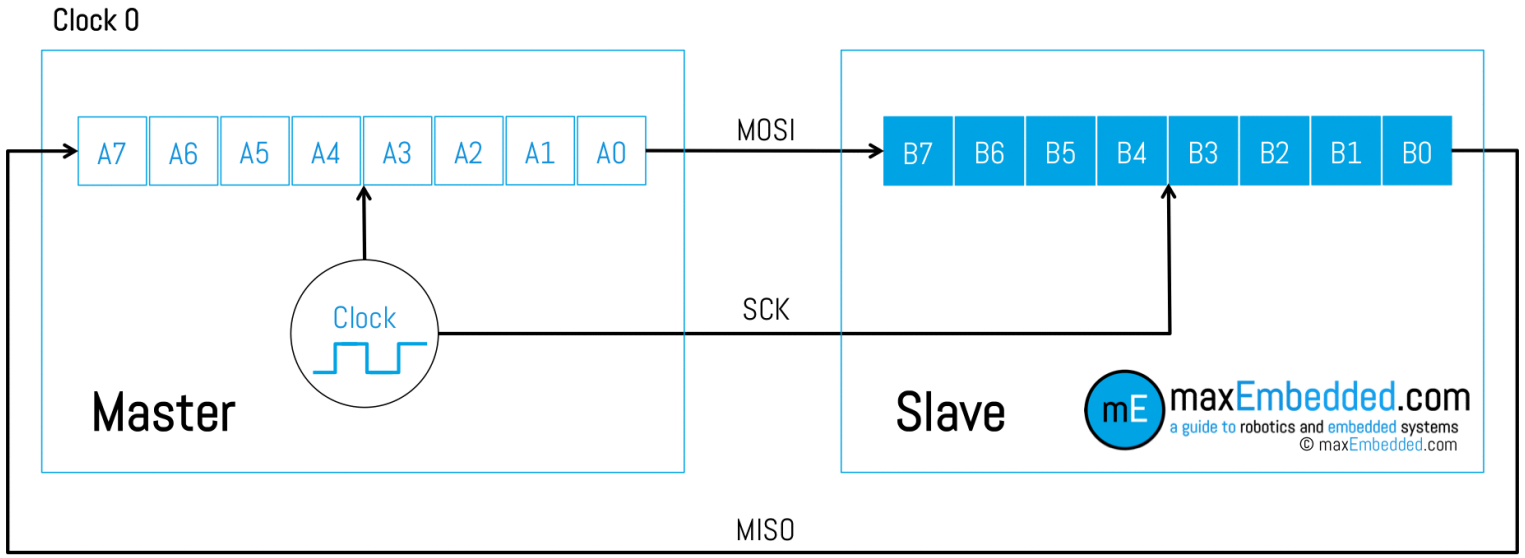
Notons que les **bus parallèle** restent grandement utilisés à l'intérieur des composants à CPU (MCU, GPP, …), voire même des SoC et ASIC.

Ici, les connexions sont suffisamment courtes et la simplicité des communication est un facteur important.





Dans une **communication série**, les données sont transmises bit par bit (un bit à la fois, un bit après l'autre).



Beaucoup de périphériques série utilisent deux registres à décalage (un pour la transmission et l'autre pour la réception).

L'étage de communication d'un composant série est bien plus complexe que celui d'un composant parallèle.

En effet, les bus internes sont parallèle. Le composant doit donc implémenter matériellement une conversion série↔parallèle, généralement à partir de registres à décalage. Un système d'horlogerie doit également être implémenté pour cadencer les échanges série.

Mais du point de vue du système, un **bus série** présente plusieurs avantages :

- Plus facile à intégrer car moins de connexions (le routage est plus simple)
- Moins de broches sur chaque composant (les CI sont plus petits, moins chers)
- Moins cher (moins de cuivre, de câbles, connecteurs plus simples à fabriquer)
- C'est plus rapide qu'un bus parallèle si les distances entre composants sont grandes

De nos jours, la grande majorité des bus (PCB, informatiques, de terrain) sont série.

Le **mode de transmission** décrit dans quelles directions l'information peut voyager.

Simplex
The information can only travel in one direction. ≈ radio broadcast

Simplex
No bus example

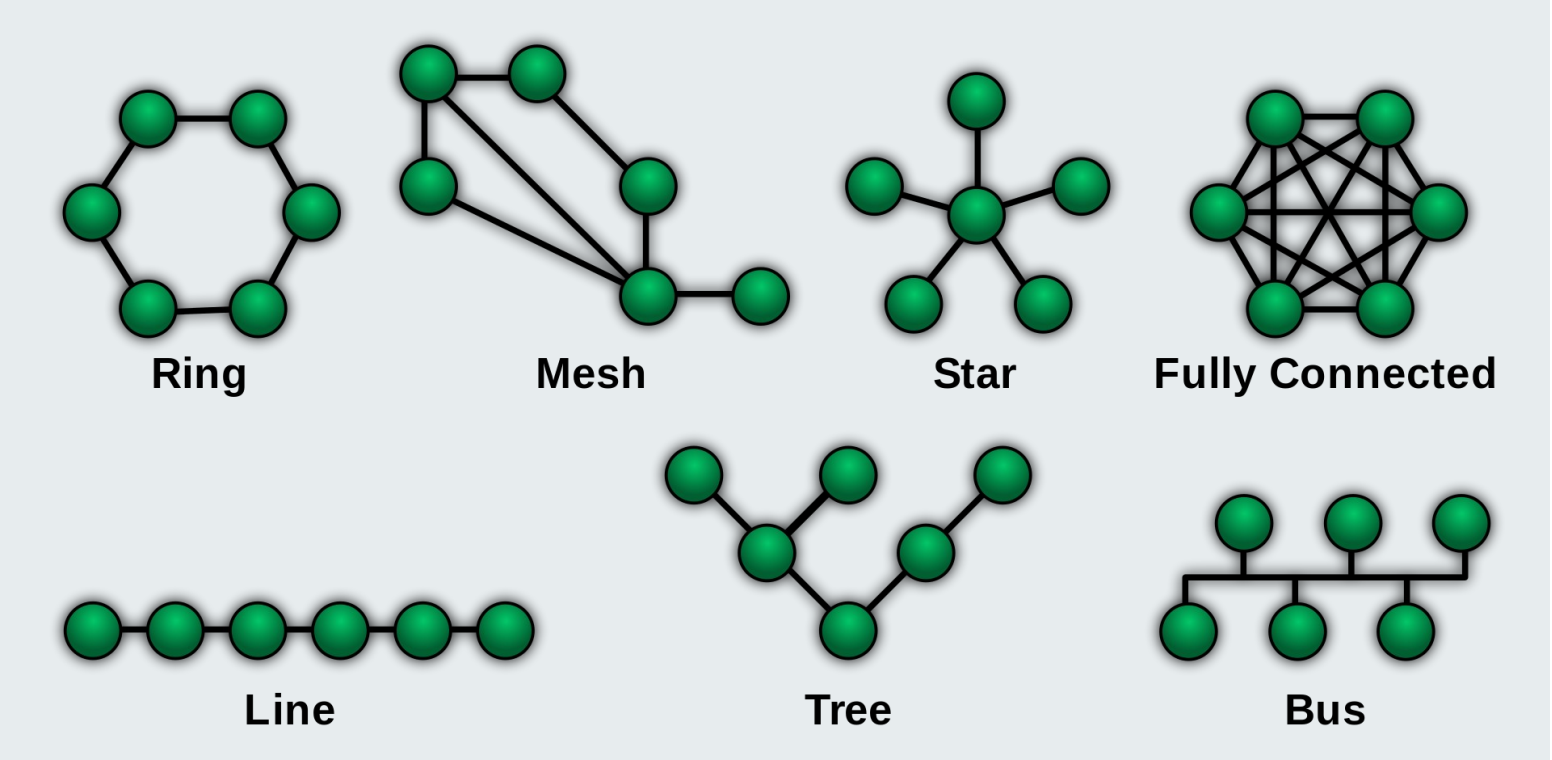
Half-duplex
The information can flow in both directions, but not at the same time. ≈ walkie-talkie

Half-duplex
1-Wire, I²C,
USB 2.0, CAN, …

Full-duplex
The information can flow in both directions, at the same time.
≈ telephone

Full-duplex
UART, SPI,
USB 3.0,
Ethernet, …

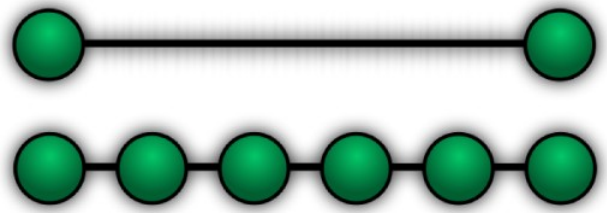
La **topologie** désigne la répartition des connexions au sein d'un réseau.



Topologie : point-à-point

La connexion **point-à-point** est la topologie la plus simple. Il s'agit de deux nœuds qui partagent un lien commun. Très simple à construire, il ne convient cependant qu'à une communication entre deux appareils. Dans l'embarqué, elle sert souvent à faire le lien entre le système et un outil de diagnostic/debug externe.

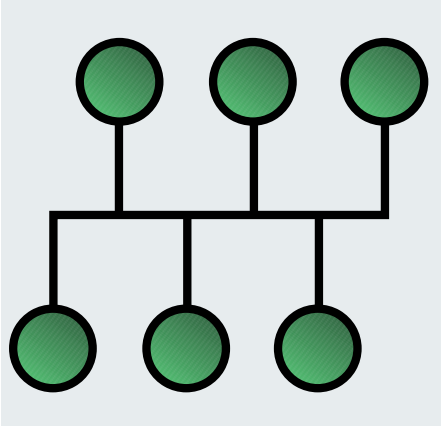
Exemples : UART, RS-232, JTAG, EODB/ODB-II, …



La connexion point-à-point est une version minimaliste de la topologie **en ligne ou en chaîne (*daisy-chain*)**. Dans ce cas, il s'agit de plusieurs connexions point-à-point, chaque nœud ayant deux voisins, chacun relié avec un canal dédié (sauf pour les nœuds aux extrémités, évidemment).

Topologie : bus

La topologie en **bus** est de loin la plus rencontrée dans les systèmes embarqués. Plusieurs nœuds peuvent se relier n'importe où sur une ligne unique et commune à tous les nœuds. C'est simple, pratique, et pas cher.



Pour éviter les collisions de données sur la ligne, le protocole doit implémenter une politique d'accès. Généralement un nœud est désigné responsable de l'accès à la ligne, sur la base du **modèle maître-esclaves**.

Exemples : SPI, I²C, USB (au niveau logique), CAN, LIN, Ethernet, ...

Choisir son bus de communication

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

La sélection d'un réseau de communication répond à plusieurs critères :

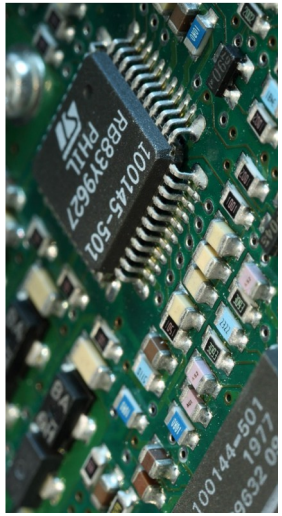
- Communauté d'utilisateurs (standards internationaux, spéc techniques, support technique, forums, …)
- Prix (royalties), licence, certifications, …
- Outils de développement (dev kits, analyseurs, outils logiciels, API, …)

Sans oublier les critères liés au produit :

- Topologie, distance, nombre de nœuds, évolutivité
- Débit et latence
- Besoin d'un ou plusieurs maîtres
- Besoin de broadcast
- Détection et/ou correction d'erreur (CRC, parité, aquittement, ré-émission, …)
- Sensibilité aux perturbations életro-magnétiques

Les réseaux (et bus) sont classés en trois catégories selon leur portée

PCB bus



- SPI
- I^2C
- Serial
- PCIe
- S-ATA
- I^2S
- ...

Inter-ICs or inter-PCBs,
Very short links.
Do not follow the OSI model,
yet can be very fast.

Computer bus

- Ethernet
- USB
- FireWire
- Serial
- ...
- HDMI
- DVI
- ...

Used as a communication interface between computer and peripherals.

Field bus
Industrial bus

- Ethernet
- CAN
- Modbus
- Lin
- Profibus
- ...

Used with sensors, actuators and vision systems.
Scale ranges from cell to production area to full-sized factory.

Le **modèle de communication** décrit la relation entre les nœuds.

En **modèle maître/esclave**, le maître est gestionnaire de l'accès au bus. Il initie chaque communication, alors que les esclaves ne peuvent que répondre. Dans cette configuration, le maître dispose d'un moyen de désigner l'esclave qui doit répondre (adresse, signal *Chip Select*, …).

Exemples : SPI, I²C, USB, LIN, Modbus, …

En **modèle client/serveur**, n'importe quel client peut effectuer une requête au serveur, qui enverra une réponse. Dans cette configuration, tous les clients connaissent l'adresse du serveur.

Exemples : c'est la base de quasiment n'importe quel service Internet (HTTP, FTP, …)

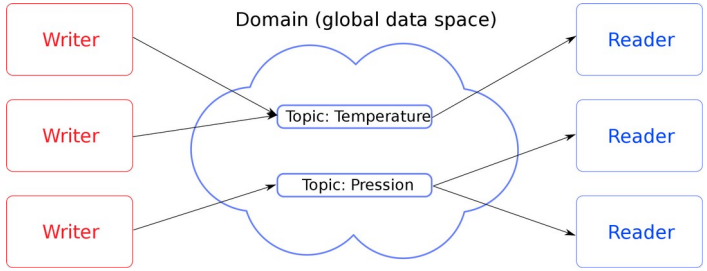
Le **modèle de communication** décrit la relation entre les nœuds.

En **modèle producteur/consommateur**, n'importe quel nœud producteur peut diffuser une information sur la ligne. Le message est reçu par tous les nœuds mais consommé que par le nœud réellement concerné.

Exemples : Ethernet, ARP, CAN, …

En **modèle publisher/subscriber**, l'abonné doit s'inscrire à un *topic*. Chaque message envoyé au *topic* par le *publisher* sera alors reçu par tous les abonnés.

Exemples : RSS, Atom,
Push server (notifications),
MQTT, Amazon MQ, …



UART ET
LIAISON SÉRIE ASYNCHRONE

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Vocabulaire

L'**UART** est un ***Universal Asynchronous Receiver-Transmitter*** : c'est un composant.

Par abus de langage, il désigne aussi le protocole de **liaison série asynchrone** pour lequel il est utilisé.

Disparu du monde informatique (le protocole USB l'a remplacé), il reste très utilisé dans le monde de l'embarqué pour sa simplicité de fabrication et d'utilisation :

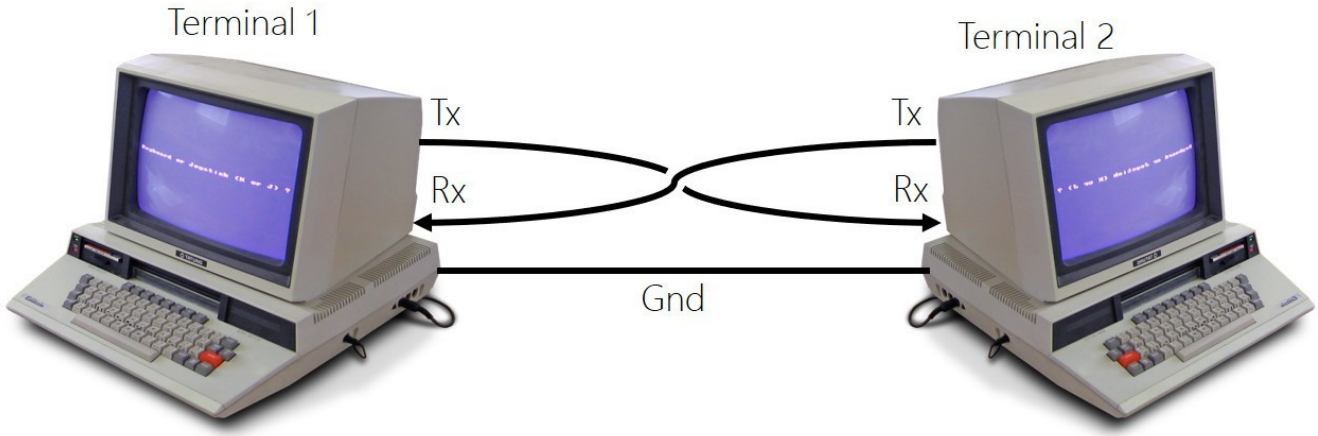
- une broche **Tx** est affectée à la transmission de données ;
- une broche **Rx** est affectée à la réception de données.

De plus, le protocole est assez malléable pour s'adapter à divers besoins.

Contrairement à la plupart des réseaux de communication rencontrés dans l'embarqué, la liaison série asynchrone UART utilise une topologie **point-à-point** (et non en bus).

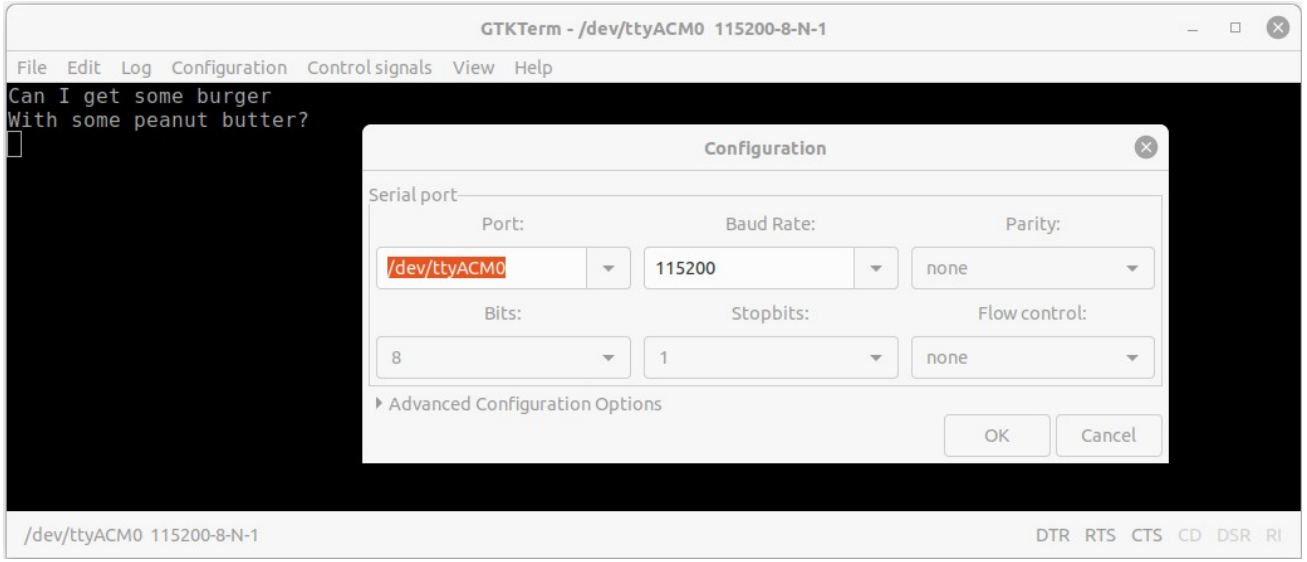
On ne peut alors relier que deux appareils, mais les branchements sont très simples car il suffit de croiser deux câbles ($Tx_1 \rightarrow Rx_2$ et $Rx_1 \leftarrow Tx_2$).

La ligne de transmission d'un appareil étant indépendante de celle de l'autre appareil, la liaison série opère en mode de transmission **full-duplex**.



De part sa topologie point-à-point l'UART est rarement utilisé entre composants d'un même PCB, mais plutôt pour relier deux systèmes entre eux.

On l'emploie notamment pour offrir un accès au système depuis un ordinateur, une sorte d'interface d'analyse ou de debug. Les données échangées sont alors de simples caractères, qui peuvent facilement être lus et transmis depuis un terminal série.



Un périphérique UART dispose donc de deux broches physiques :

- Rx pour la réception (branché sur la Tx de son homologue)
- Tx pour la transmission (branché sur la Rx de son homologue)

Dans l'embarqué, les signaux Rx et Tx utilisent les niveaux de tension TTL :

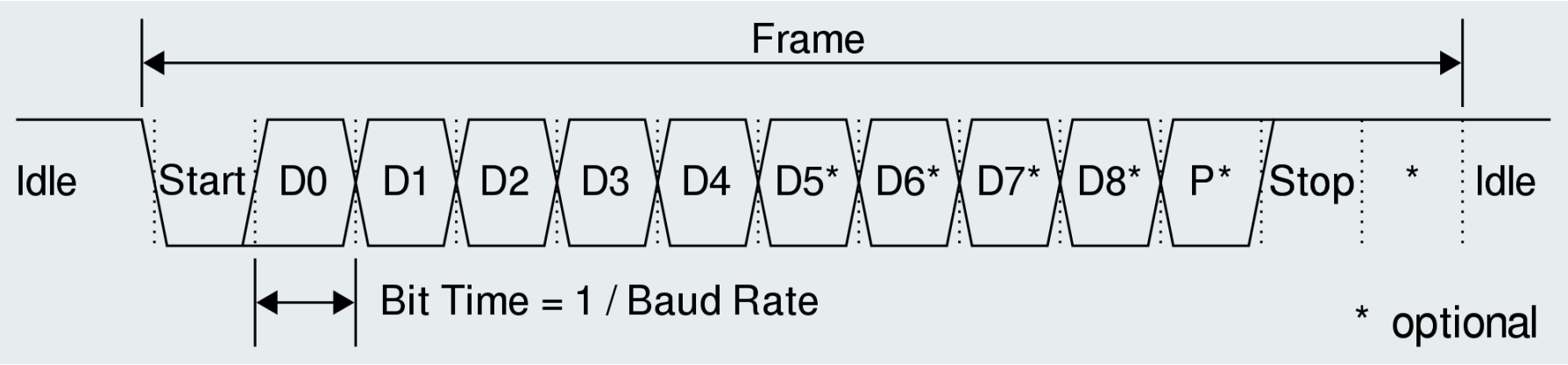
- 0 V pour un niveau bas logique
- +5 V pour un niveau haut logique
- À l'état de repos (*idle state*), la broche Tx impose un niveau haut logique.

D'autres standards de liaison série asynchrone existent, parmi lesquels on peut citer le RS-232 et le RS-485. Ils redéfinissent la couche Physique du modèle OSI : niveaux de tension, codage, connecteur, paire différentielle, longueur des câbles, … mais respectent le format de trame décrit page suivante.

Le protocole fixe la structure des trames échangées, même si elle est assez flexible.

Une trame est constituée de plusieurs champs :

- 1 bit de start (niveau bas logique, pour contraster avec l'état de repos)
- 5 à 9 bits de données (bit de poids faible en premier)
- 1 bit de parité, optionnel, servant à la détection d'erreur
- 1 ou 2 bits de stop (niveau haut logique)



SPI

Serial Peripheral Interface

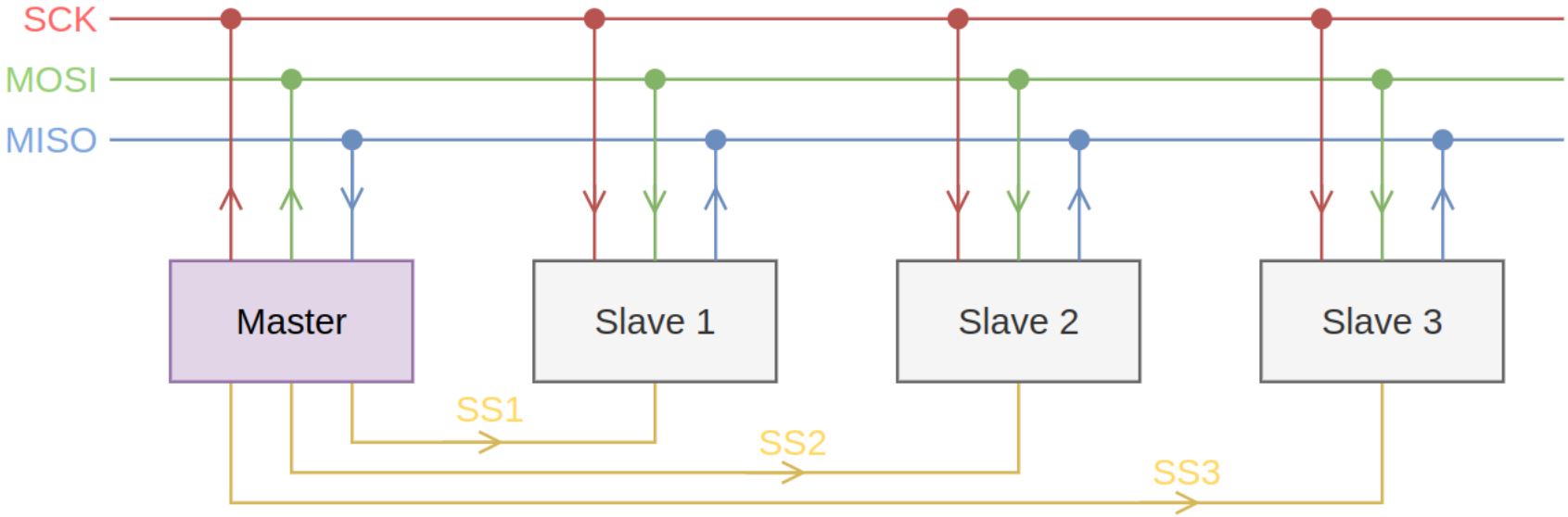
ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Le **SPI (*Serial Peripheral Interface*)** est très répandu au sein des systèmes embarqués.

Il s'agit d'un bus de communication **série**, **synchrone**, **full-duplex**, **maître-esclaves**.

Proposé dans les années 80 par Motorola, aucune norme ne vient pourtant définir le SPI, ce qui laisse une certaine marge de manœuvre (ou un flou artistique).



Signaux

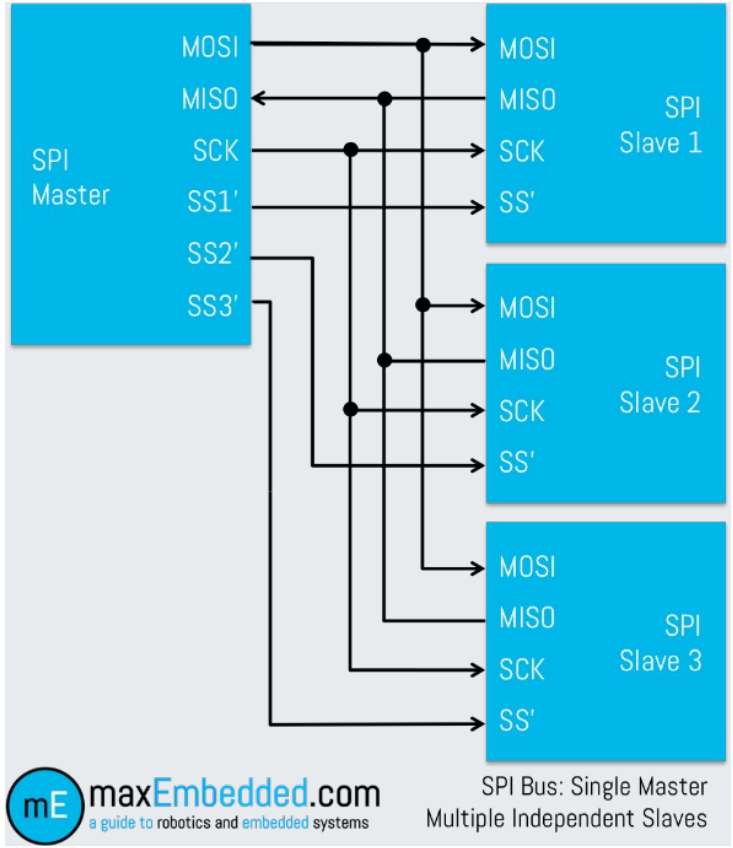
Le bus SPI est composé de trois lignes communes :

- **SCK (*Serial Clock*)**, contrôlée par le maître ;
- **MOSI (*Master Out, Slave In*)**, contrôlée par le maître ;
- **MISO (*Master In, Slave Out*)**, contrôlée par un esclave.

Il y a également une ligne propre à chaque esclave :

- **$\overline{\text{SS}}$ (*Slave Select*)**, contrôlée par le maître pour désigner l'esclave interlocuteur.

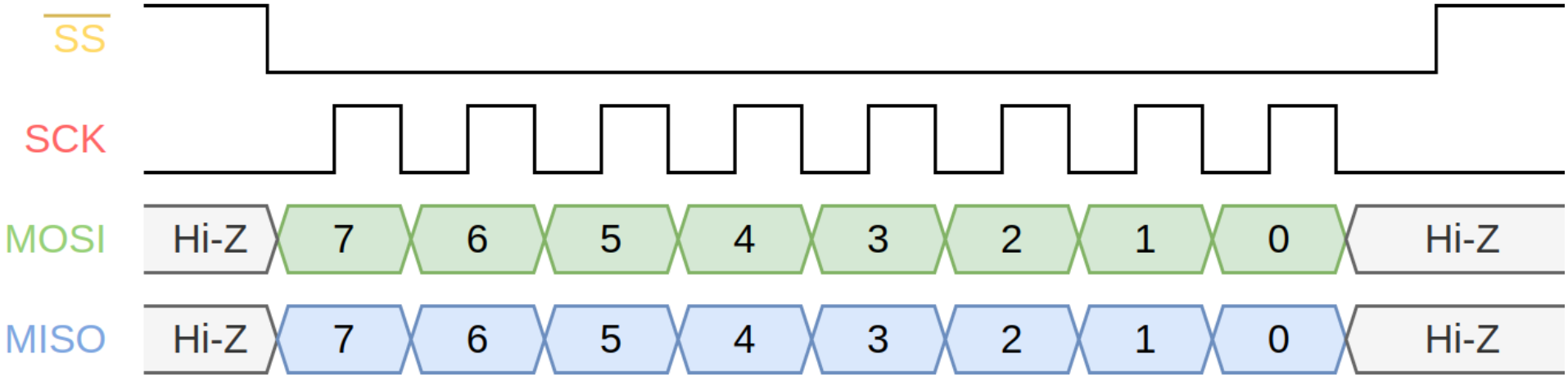
Note : La connotation historique du terme « esclave » a amené les acteurs du marché à renommer les signaux. On peut alors rencontrer plusieurs dénominations différentes : CS (*Chip Select*), SDI/SDO (*Serial Data In/Out*), *Main/Sub*, *Controller/Peripheral*, … Le SPI n'étant pas normalisé, les acteurs poussent chacun de son côté pour mettre en avant sa propre terminologie.



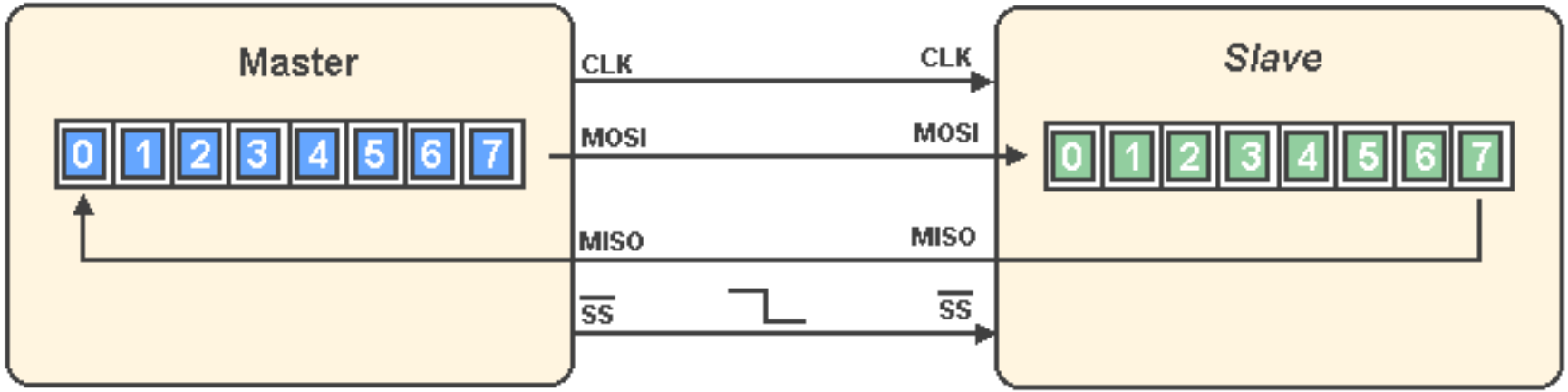
Format de la trame

Dans la version la plus répandue du SPI, le maître désigne l'esclave concerné par l'échange en abaissant son signal $\overline{SS}$.

Ensuite le maître et l'esclave fournissent un bit à chaque front descendant d'horloge (front descendant de $\overline{SS}$ pour le premier bit), et chaque bit sera lu sur le front montant d'horloge.



Matériellement, le contrôleur SPI du composant peut se résumer à un simple registre à décalage. Il est donc simple à concevoir et économe en énergie.



Notons qu'il n'y a pas de bit de start, de stop, de détection d'erreur, … Les bits transmis sont uniquement les données utiles.

Configurations d'horloge

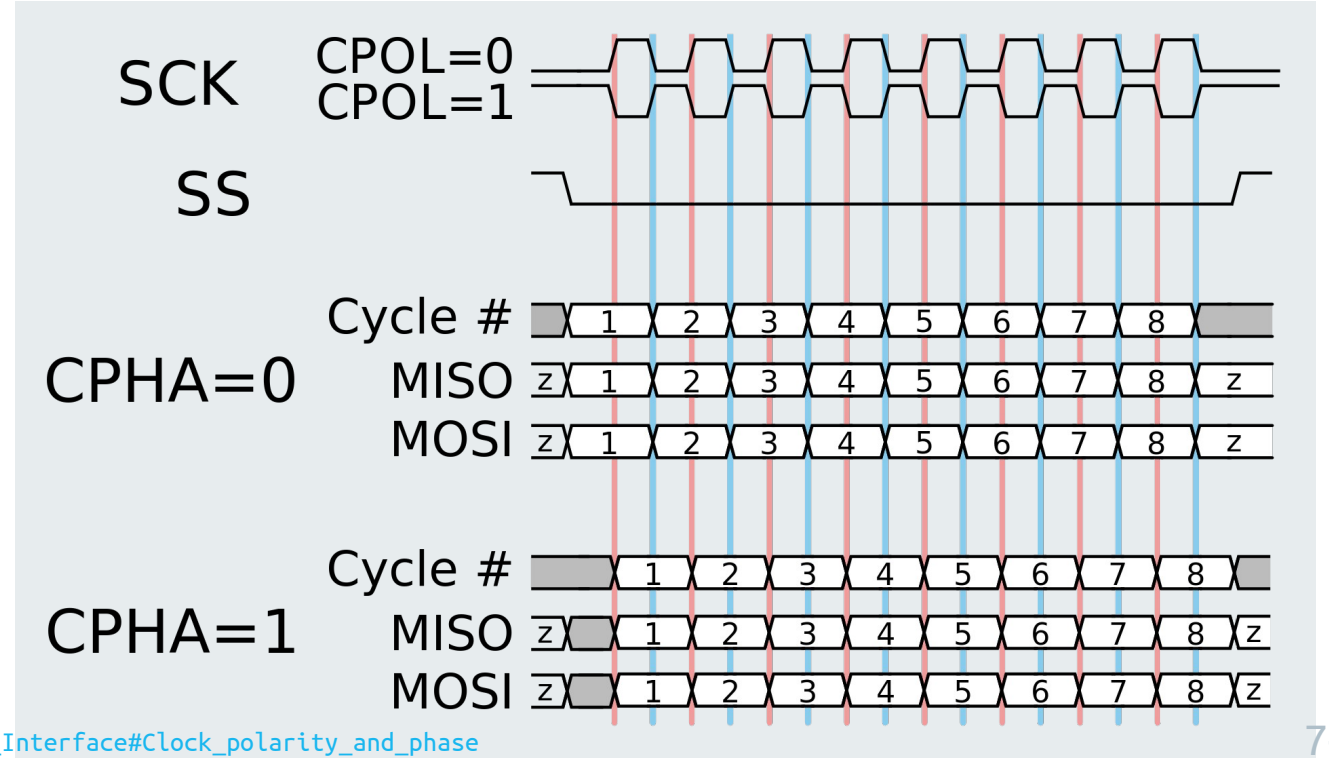
Il existe 4 différentes configurations d'horloge, indiquées par la CPOL (*Clock Polarity*) et la CPHA (*Clock Phase*).

Les modes 0 et 3 sont compatibles entre eux, les modes 1 et 2 sont compatibles entre eux.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

CPOL représente la polarité du signal d'horloge au repos.
'0' = état de repos au niveau bas
'1' = état de repos au niveau haut

CPHA indique le front d'horloge sur lequel le bit sera échantillonné (lu) :
'0' = premier front d'horloge,
'1' = deuxième front



https://en.wikipedia.org/wiki/Serial_Peripheral_Interface#Clock_polarity_and_phase

Avantages et inconvénients

Pros

Débit important (*full-duplex*, débit utile = débit réel)

→ 1 Mbit/s à 100 Mbit/s

Charge utile à taille variable (nombre de bits)

→ généralement 8 bits, mais peut faire du 16-24-32

Contrôleur peu complexe (registre à décalage = faible consommation)

Pas d'arbitrage

Signaux uni-directionnels (isolation galvanique facile)

Cons

Gourmand en broches : 3 + n (n = nombre d'esclaves)

Pas d'acquittement (le maître peut parler dans le vide)

Ni détection ni gestion des erreurs

Pas de contrôle de flux

Multi-maître impossible

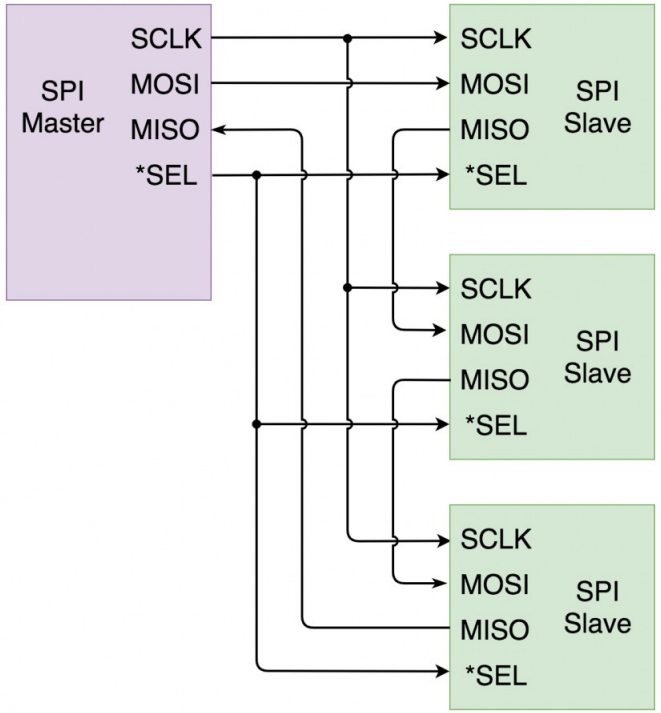
Communication sur de courtes distances

Variantes

Il existe aussi plusieurs variantes de SPI.

On peut parler de la configuration *daisy-chain*, où la topologie en bus n'est plus utilisée mais remplacée par une topologie en ligne. Chaque esclave retransmet la donnée à son successeur. Longueur totale de MOSI plus courte, et un seul signal de SS.

Il existe aussi le bus *three-wire SPI*, qui remplace les deux lignes MOSI et MISO par une seule ligne de données bidirectionnelle. Le protocole devient donc *half-duplex*, mais permet d'économiser un fil. On le retrouve dans des applications basse-consommation.



I²C

Inter-Integrated Circuit

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Description

Le bus **I²C (*Inter-Integrated Circuit*)** a été développé par Philips Semiconductors (aujourd'hui NXP) en 1992.

Il s'agit d'un bus de communication **série**, **synchrone**, **half-duplex** et en topologie **maître-esclaves**. Contrairement au SPI, l'I²C offre la possibilité d'avoir plusieurs maîtres sur le bus.

L'utilisation du bus est libre, mais tous les composants doivent disposer d'une adresse, à réserver (payer) auprès de NXP.

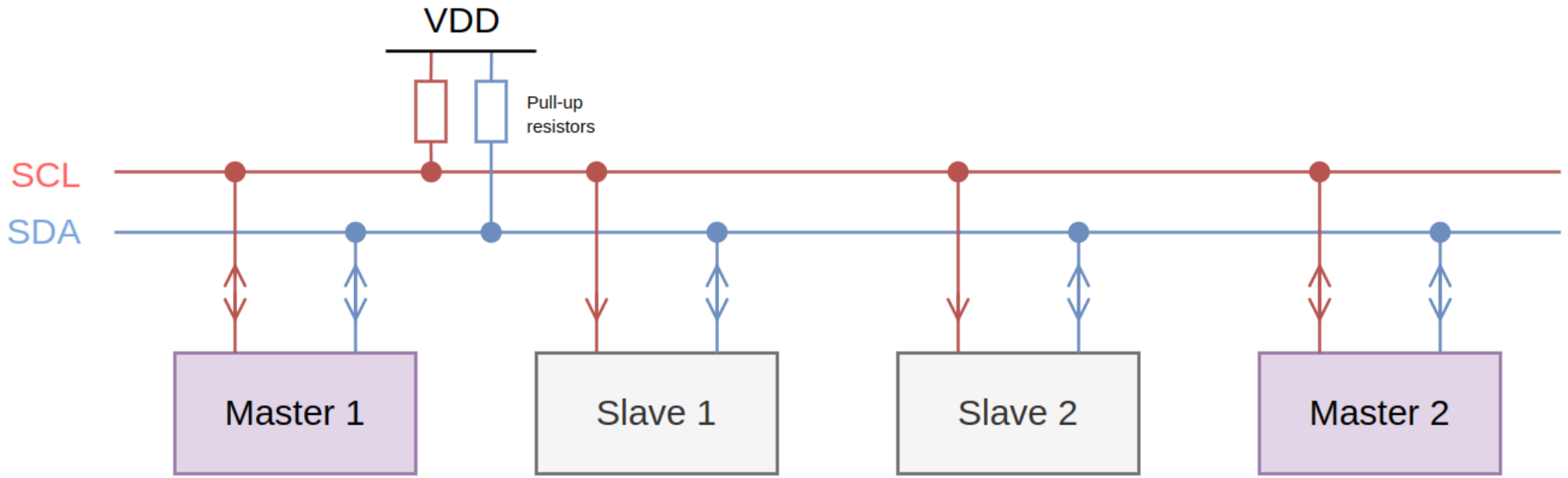
Les débits de transmission sont normalisés :

100 kbit/s (standard mode), 400 kbit/s (fast mode), 1 Mbit/s (Fast plus mode), 3.4 Mbit/s (High-speed mode), 5 Mbit/s (Ultra-fast mode).

Le bus I²C ne contient que deux fils :

- **SCL (*Serial Clock*)**, ligne d'horloge en départ d'un des maîtres, à destination des autres composants ;
- **SDA (*Serial Data*)**, ligne de données bidirectionnelle.

Ces deux lignes doivent posséder une résistance de *pull-up* pour forcer un état de repos au niveau logique haut.

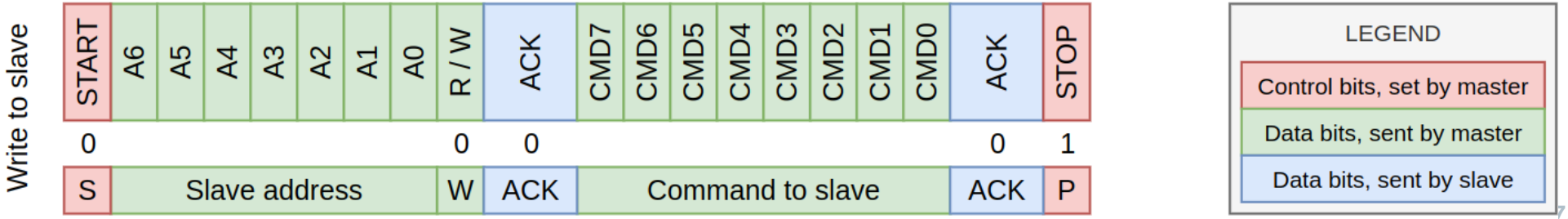


Format de la trame

La ligne SDA étant bidirectionnelle, le maître et l'esclave désigné doivent respecter une certaine chorégraphie temporelle.

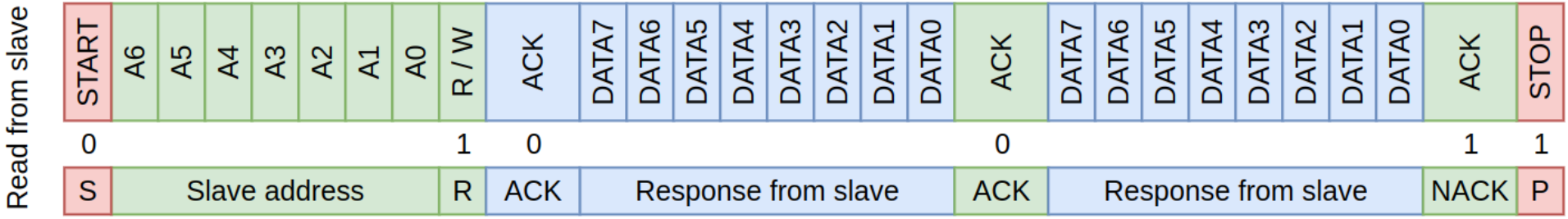
La maître initie la conversation avec un bit de start (niveau bas), 7 bits d'adresse pour sélectionner un esclave et un bit pour le mode d'accès (R/$\overline{W}$). Si un esclave reconnaît son adresse, alors il acquitte en imposant un niveau logique bas sur la ligne.

Dans le cas d'une écriture (bit R/$\overline{W}$ = 0) du maître vers l'esclave, le premier continue la transmission de données (8 bits), et attend l'acquittement (niveau bas) de l'esclave. Cette opération se répète jusqu'à ce que le maître envoie un bit de stop (niveau haut).



La ligne SDA étant bidirectionnelle, le maître et l'esclave désigné doivent respecter une certaine chorégraphie temporelle.

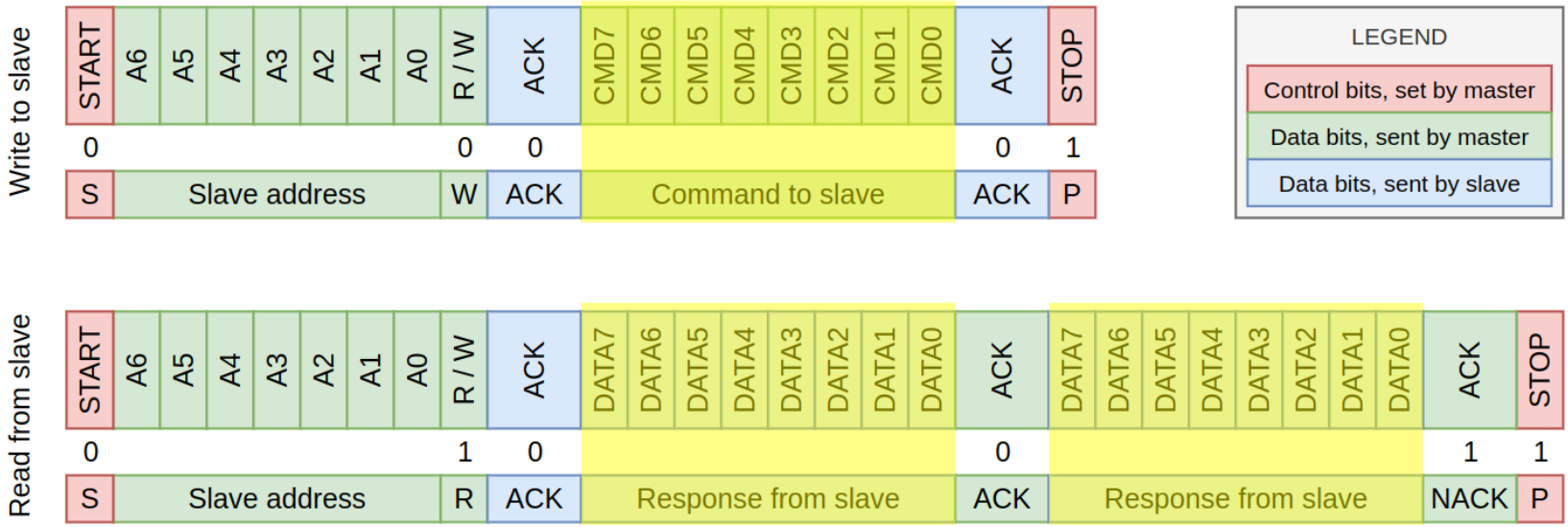
Dans le cas d'une lecture (bit R/$\overline{W}$ = 1) du maître depuis l'esclave, ce dernier effectue une transmission de données (8 bits), et le maître acquitte (niveau bas) l'octet reçu. Cette opération se répète jusqu'à ce que le maître envoie un bit de stop (niveau haut).



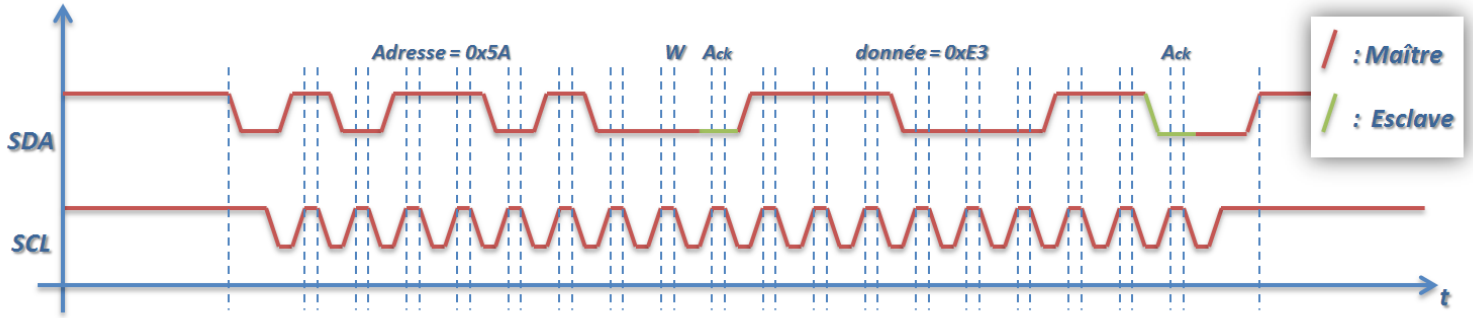
Note : la ligne SCL est également censée travailler de concert avec la ligne SDA, mais elle n'est pas représentée ici pour des raisons de simplicité et lisibilité.

La sélection de l'esclave se fait donc par bits d'adresse dans la trame (et non par une ligne dédiée). Il y a également des bits d'acquittement, ce qui permet de détecter l'absence (ou la non-compréhension) d'un esclave sur le bus.

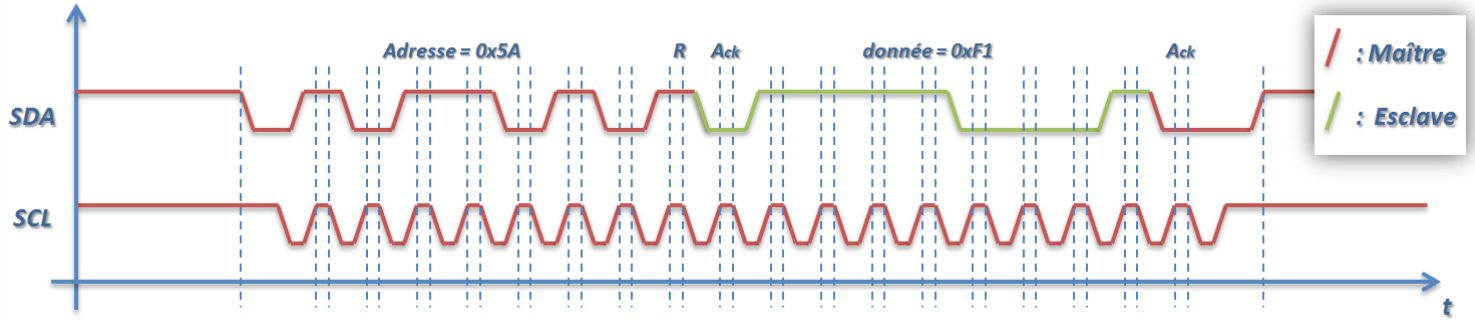
La contre-partie est que le débit utile des données est bien inférieur au débit réel des bits sur le bus.



Exemple : le maître effectue la requête 0xE3 à l'esclave d'adresse 0x5A.

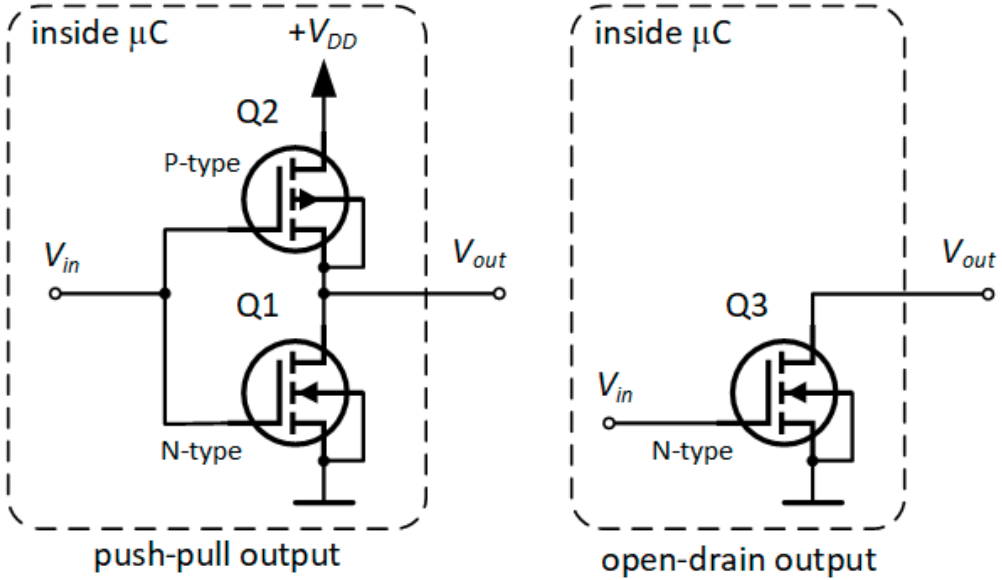


Le maître lit ensuite la réponse de l'esclave, qui renvoie la donnée 0xF1.



Les broches SCL et SDA des composants I²C sont en **drain ouvert (open drain output),** c'est à dire un seul transistor N-MOS.

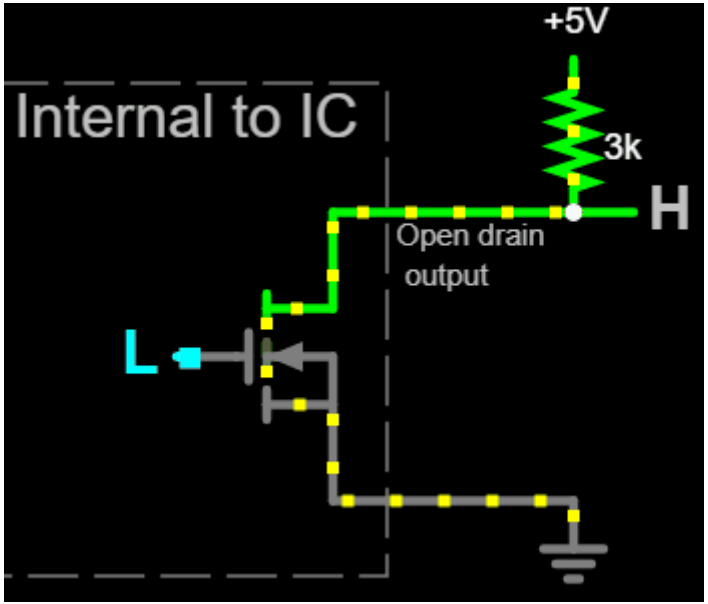
À titre de comparaison, les sorties numériques des circuits intégrés sont généralement des push-pull CMOS (*Complementary* MOSFET). Cela permet d'imposer un niveau logique bas (Q1 passant), un niveau logique haut (Q2 passant), voire un troisième état appelé haute-impédance (*Hi-Z*) (Q1 et Q2 bloqués).



Les lignes SCL et SDA sont chacune associées à une **résistance de pull-up**.

Quand, pour tous les composants, le transistor en drain ouvert est bloqué, alors la ligne est « tirée » au niveau haut logique. C'est son état de repos.

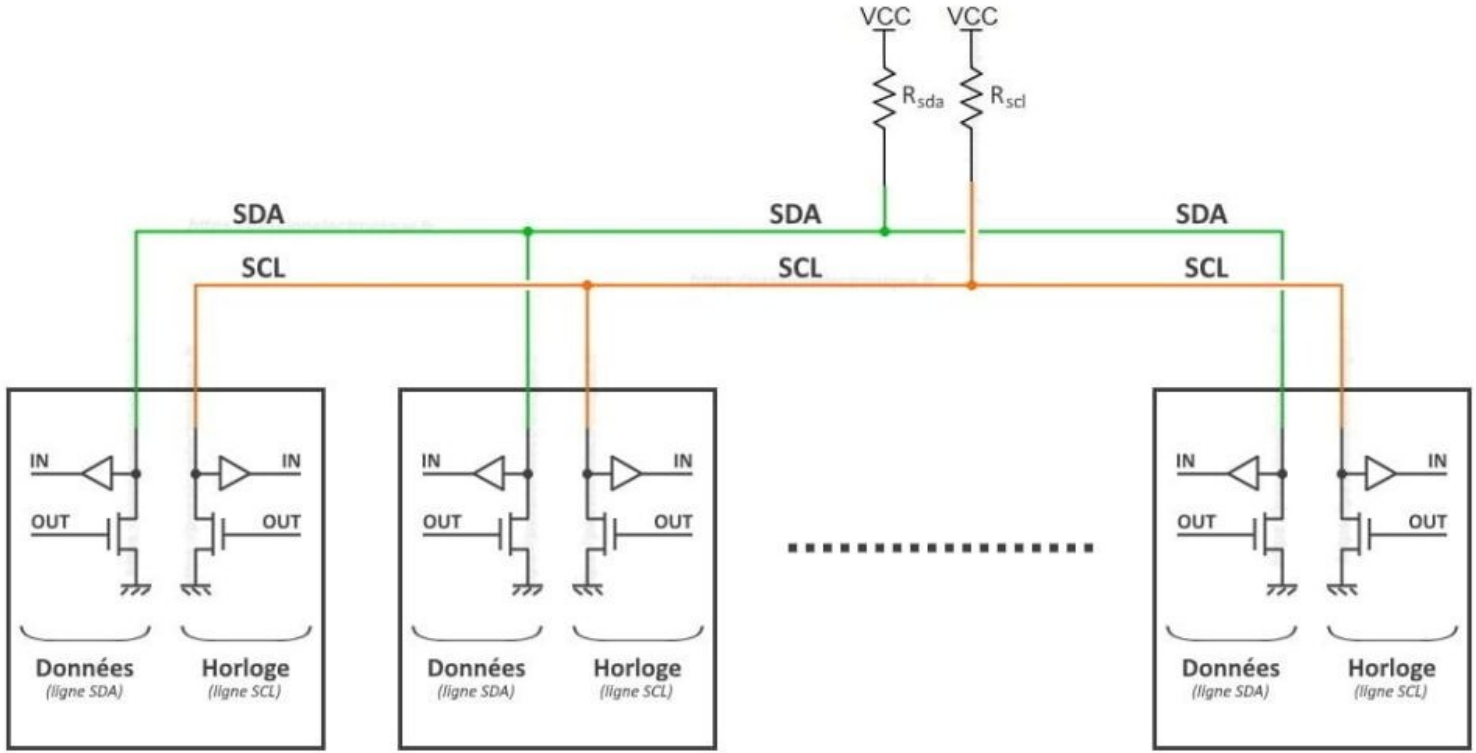
Dès lors que l'un des composants passe son transistor en passant, alors la ligne se retrouve directement reliée à la masse et un niveau bas logique est imposé.



Cela signifie que si deux composants veulent écrire en même temps sur la ligne, l'un un '0' et l'autre un '1', alors le niveau bas sera dominant (le niveau haut étant alors récessif).

Une logique d'arbitrage doit également être mise en place lorsqu'un conflit d'écriture est détecté.

Bien entendu, en plus de l'aspect écriture sur la ligne, un étage buffer d'entrée permet d'assurer la lecture du niveau logique des lignes SCL et SDA.



Avantages et inconvénients

Pros

Nombre réduit de fils (SCL + SDA)

Payload de taille variable

Acquittement possible

→ détection d'esclave

→ détection d'erreur

Multi-maître possible

Cons

Débit utile faible

→ Débit réel faible (< 3.4 Mbit/s)

→ Overhead important (start, adresse, ack, stop, …)

Uniquement sur de courtes distances

Contrôleur complexe

Logique d'arbitrage à implémenter en multi-maître

Plage d'adressage payante (NXP)

ALLER PLUS LOIN

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Synthèse UART – SPI - I²C

	UART	I2C	SPI
Nbre de fils à prévoir (hors masse/alimentation)	2 fils (RX et TX)	2 fils (SDA et SCL)	3 fils (SCK, MOSI, et MISO) + 1 fil suppl. par esclave
Données	5 à 9 bits	8 bits	8 bits, typiquement
Adressage	Aucun (connexion unique entre 2 éléments)	Sur 7 bits, encodé à même le message envoyé	Physique (1 fil par esclave à piloter)
Vitesses de transmission (les plus utilisées, donc non exhaustif)	1200 à 115200 bits/s (9600 bauds étant la vitesse la plus commune)	100k à plusieurs Mbits/s 100 kbits/s = "Standard Mode" 400 kbits/s = "Fast Mode" 1 Mbits/s = "Fast Plus Mode" (au delà : high speed, ultra fast mode)	De 1 à 20 MHz de fréquence d'horloge, classiquement (4 MHz de base, par ex, pour un Arduino tournant à 16 MHz)
Distance max de transmission	Jusqu'à quelques mètres (et jusqu'à plusieurs dizaines de mètres avec le protocole RS-232)	Jusqu'à quelques mètres	Jusqu'à quelques mètres
Accusé de réception ("ACK" en anglais, pour "acknowledgment")	Aucun	1 bit d'ACK ou NACK, après chaque paquet de données reçu ou envoyé	Aucun
Résistances pull-up	Aucune nécessaire	1 sur SDA et 1 sur SCL (valeurs typiques = 4,7 k-ohms chacune)	Aucune nécessaire

Très variable attention, selon la vitesse de transmission, et la qualité des lignes. À noter que ces distances peuvent être étendues, avec des buffers.

Premier site à partager pour une compréhension rapide des bus de communication évoqués :

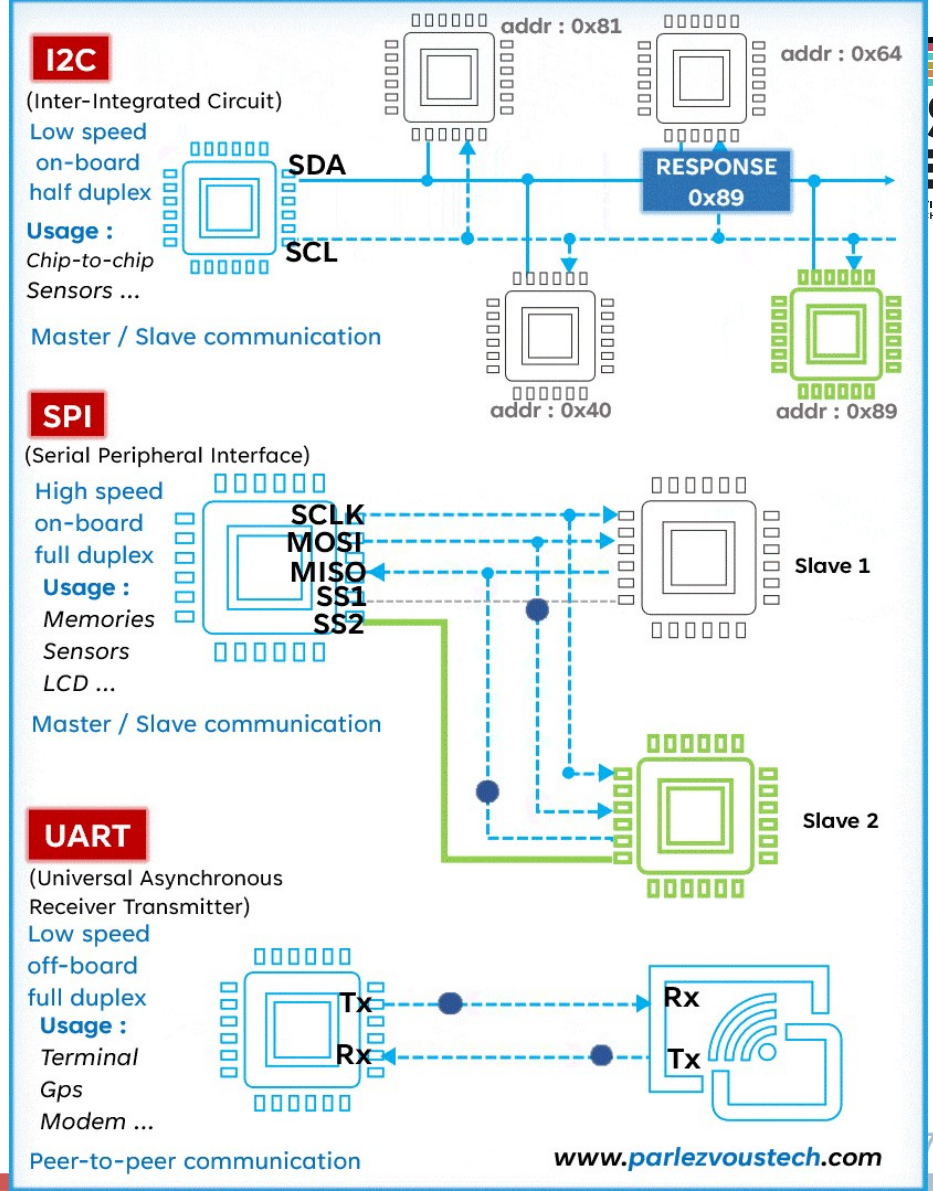
https://passionelectronique.fr/liaisons-series-uart-i2c-spi/

Synthèse UART – SPI - I²C

Second site à partager pour une compréhension rapide des bus de communication évoqués :

https://www.parlezvoustech.com/en/comparaison-protocoles-communication-i2c-spi-uart/

(les GIFs du site sont intéressants).



D'autres protocoles mériteraient d'être abordés ici, par exemple :

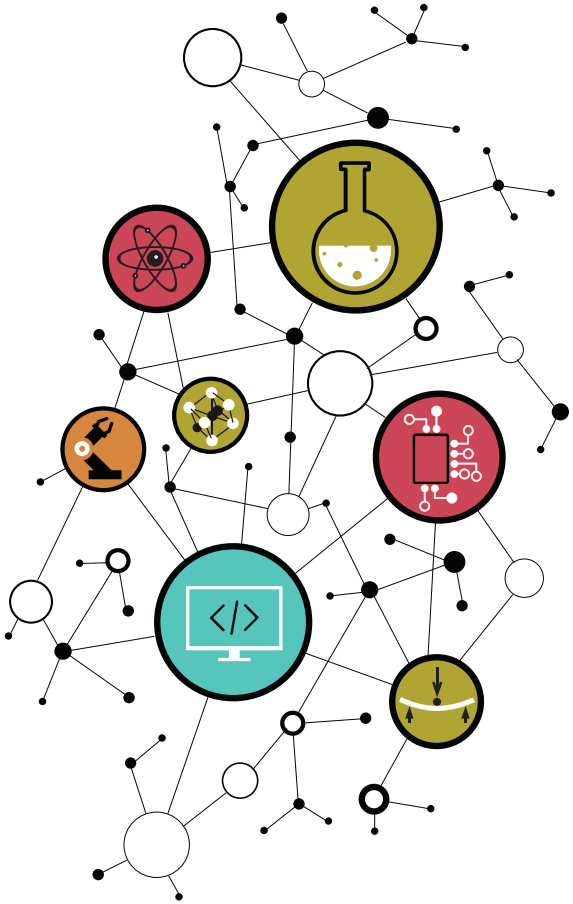
- 1-wire , bus PCB utilisé dans des applications bas-débit et basse-consommation ;
- USB (*Universal Serial Bus*) , bus informatique (et non bus PCB) aujourd'hui incontournable.

On pourrait également discuter d'autres protocoles de communication ne relevant pas du bus mais de la radio, fréquemment rencontrés dans l'embarqué :

- RFID, NFC : des protocoles à champ proche (~ 1-10 cm)
- Bluetooth, ZibBee, Wi-Fi : des protocoles à légèrement plus longue portée (~ 10-100 m)
- LoRa : un protocole à longue portée (~ 1-10 km), faible débit (250 à 11000 bit/s), spécialisé pour l'IoT.

Vous aurez un cours de réseaux (orienté TCP/IP) en deuxième année.

CONTACT



Dimitri Boudier – PRAG ENSICAEN

dimitri.boudier@ensicaen.fr

Avec l'aide précieuse de :

- Hugo Descoubes (PRAG ENSICAEN)