# Capteurs numériques

## 1. Comparaison des Bus de Communications

| Caractéristique | UART | I2C | SPI | USB | CAN |
|---|---|---|---|---|---|
| Type de bus | Asynchrone | Synchrone | Synchrone | Synchrone | Synchrone |
| Mode de communication | Série, unipolaire | Série, unipolaire | Série, unipolaire | Série, différentiel | Série, différentiel |
| Nombre de fils | 2 (Rx, Tx) + GND | 2 (SDA, SCL) + GND | 3+ (MISO, MOSI, SCK, SS) + GND | 4 (Vcc, GND, D+, D-) | 2 (CAN_H, CAN_L) + GND |
| Vitesse de transfert | Jusqu'à 5 Mbps | 100 kbps à 5 Mbps (High-speed mode) | Jusqu'à 10 Mbps ou plus | Jusqu'à 20 Gbps (USB 4) | Jusqu'à 1 Mbps |
| Topologie | Point à point | Multi-maître, multi-esclave | Maître-esclave | Hôte-périphérique | Réseau multipoint |
| Complexité | Faible | Moyenne | Moyenne | Élevée | Moyenne |
| Adressage | Non applicable | 7 ou 10 bits | Par sélection de l'esclave (SS) | Par adressage | Non applicable |
| Support de détection d'erreur | Parité, contrôle de flux (facultatif) | Non | Non | CRC, retransmission, contrôle de flux | CRC, détection d'erreurs de trame |
| Coût | Faible | Faible à moyen | Faible à moyen | Moyen à élevé | Faible à moyen |

Voici un exemple d'application pour chaque bus de communication :

1. UART (Universal Asynchronous Receiver-Transmitter):

Communication entre un microcontrôleur et un module GPS, modem radio, ou bien entre deux microcontrôleurs qui veulent pouvoir agir en maitre.

2. I2C (Inter-Integrated Circuit) :

Capteurs et périphériques à faible vitesse et débit de data : l'I2C est couramment utilisé pour la communication entre un microcontrôleur et des capteurs (Température, pression etc.)

3. SPI (Serial Peripheral Interface) :

Cartes SD et écrans TFT : Le SPI est souvent utilisé pour la communication entre un microcontrôleur et des périphériques à plus grande vitesse, tels que les cartes mémoire SD pour le stockage de données ou les écrans TFT pour l'affichage d'images et de vidéos. Pour les bus les plus rapides pour l'image on se dirige ensuite vers des bus LCD, MIPI, DSI…

4. USB (Universal Serial Bus) :

Périphériques d'ordinateur : USB est couramment utilisé pour connecter des périphériques externes à un ordinateur. Il est aussi utilisé pour fournir l'alimentation en plus de transporter la donnée

5. CAN (Controller Area Network) :

Réseaux de véhicules et automatisation industrielle : Le CAN est largement utilisé dans les systèmes embarqués de véhicules pour la communication entre les différents modules électroniques sans nécessiter un maitre central. Le CAN est également employé dans les applications pour connecter divers capteurs, actionneurs, où la fiabilité et la résistance aux interférences sont cruciales.

# 2. Interfacer un capteur SPI

Page moodle du cours interfacer un capteur SPI

https://moodle.insa-toulouse.fr/mod/page/view.php?id=39145

# 3. Interfacer un capteur I2C

Page moodle du cours interfacer un capteur I2C

https://moodle.insa-toulouse.fr/mod/page/view.php?id=39144

# 4. Le protocole UART

Page moodle du cours UART

https://moodle.insa-toulouse.fr/pluginfile.php/172074/mod_resource/content/1/UART.pdf

# 5. Pourquoi cela ne fonctionne pas ?

Lors de la mise ne place d'un bus, souvent le fonctionnement n'est pas immédiat. Je liste ici quelques pistes les plus fréquentes de raisons de non-fonctionnement d'un bus :

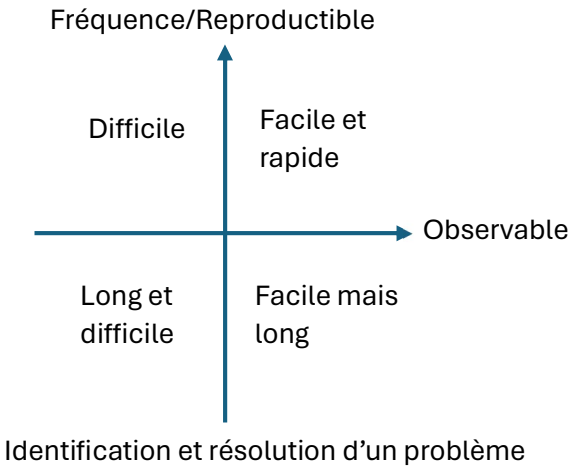
- **Câblage mal réalisé :** Pin du µcontrôleur non disponible pour la fonction voulue, il faut espérer que les IOs soient remappables
- **Périphérique excluant un autre :** pas possibilité d'utiliser UART1 et SPI1 en même temps, le périphérique est partagé entre les deux. Et souvent la solution de repli de passer sur UART2 ou SPI2 n'est pas disponible car les pins ne correspondent pas => il faut revoir le routage
- **Alimentation du capteur non fournie :** Plus souvent qu'on ne le pense => le bus marchotte mais ne fonctionne pas proprement, comportements erratiques. Bien vérifier que le composant est alimenté.
- **Mode de fonctionnement non compatible :** Si le composant est en Stop Mode par exemple il se peut que son bus soit désactivé (typique sur les drivers de moteur si les EN sont désactivés, le composant est en sleep mode et le bus ne répond pas)
- **Vitesse de transmission (Baud Rate) incorrecte :** Si les dispositifs ne sont pas configurés pour la même vitesse de transmission, les données ne seront pas correctement interprétées. Donne quand même des choses mais non cohérentes
- **Configuration de l'horloge :** Une mauvaise configuration de l'horloge peut entraîner une désynchronisation des données => donner une source d'horloge cohérente
- **Paramètres du bus incorrects :** UART : les bits de stop, la parité SPI : Mode, fréquence … I2C : Unicité de l'adresse, valeur pull up…
- **Masse non commune :** Il est impératif de partager la masse entre les différents objets qui communiquent par un bus
- **Tensions incompatibles :** Des différences de tension entre les dispositifs peuvent causer des problèmes de communication, on peut placer un adaptateur de niveau entre les composants
- **Bruit électrique :** Le bruit sur les lignes d'alimentation ou de données peut corrompre les signaux.
- **Erreurs dans le code :** Des bugs ou des erreurs logiques dans le code qui gère la communication peuvent aussi être la cause.
- **Capacité de charge :** Dépasser la capacité maximale de charge du bus peut dégrader les signaux.
- **Distance :** Une distance excessive entre les composants peut augmenter l'atténuation du signal et les interférences.

En général la stratégie est de

# 6. Debugger un BUS de communication

La stratégie pour le degug des bus de communications est de tester

1. Les points rapides à tester
2. Les points les plus plausibles
3. Le reste




Sur un bus de communication il y a un moyen rapide de rendre un problème observable, il s'agit de placer un analyseur logique sur les lignes du bus.

Un analyseur logique capture et affiche les signaux électriques passant à travers les bus de communication, ce qui permet aux utilisateurs de voir l'activité sur ces bus et de comprendre ce qu'il se passe sur le bus. Parfois, même savoir qu'il n'y a pas d'activité sur le bus est une indication précieuse pour la mise au point du software.

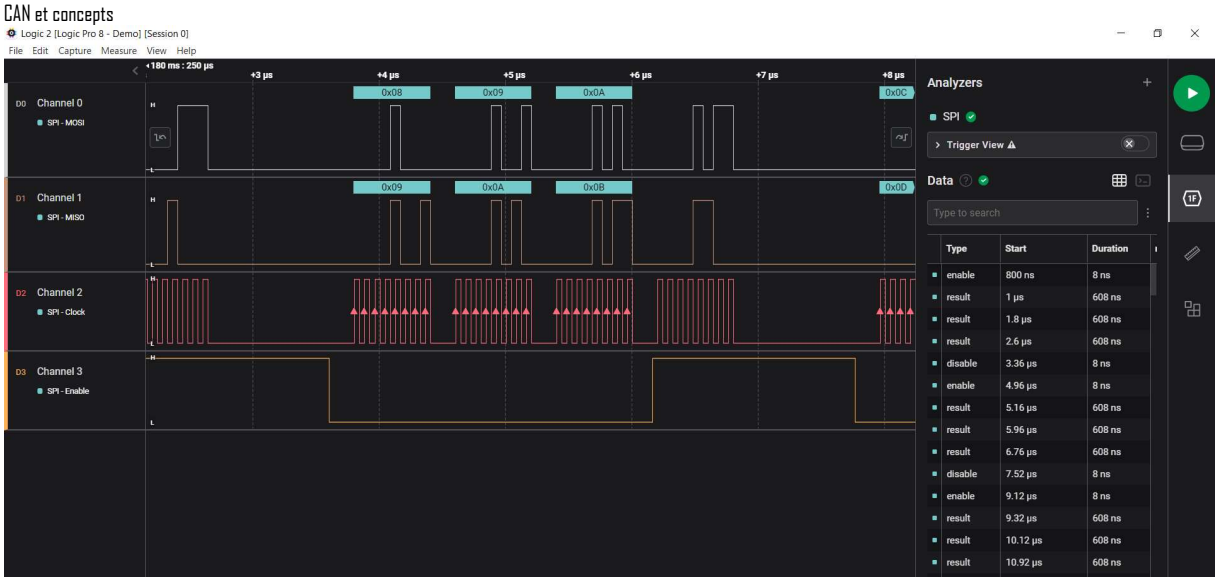



Image d'une capture d'un analyseur logique (SPI)

Voyez-vous le problème ?

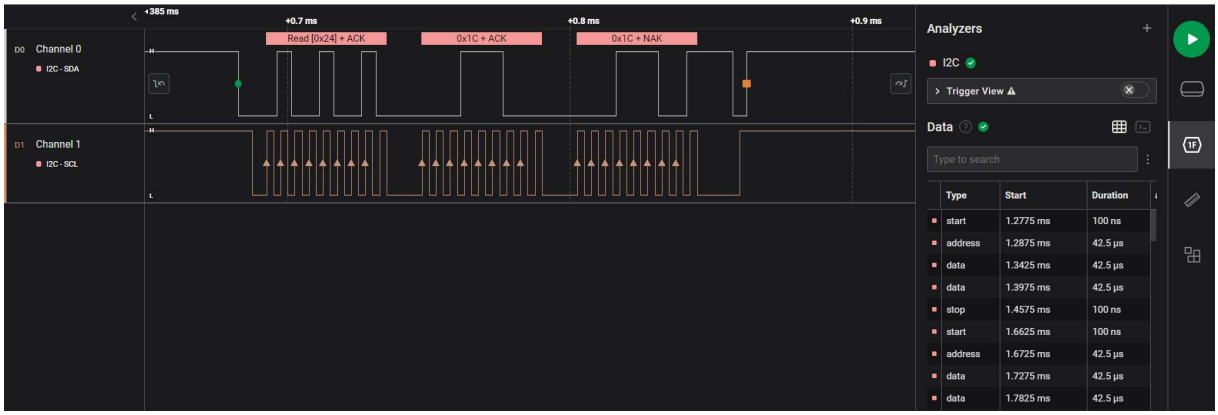



Image d'une capture d'un analyseur logique (I2C)

## Point d'attention

Un analyseur logique, comme tout outil de mesure peut déséquilibrer un système et introduire du bruit sur un bus. Souder des fils sur la carte pour extraire les lignes du bus introduit des antennes et des boucles de courant. De plus cela ajoute un condensateur parasite. Dans les phases de débug, on réduira au maximum la fréquence d'échange des données.

# 7. Exemples de problèmes rencontrés

## I2C

Sur un produit, plusieurs capteurs sont placés sur un bus I2C. La lecture des IDs de chaque composant est concluante, et les mesures remontent bien de ces capteurs. L'application est développée et déployée chez le client. Le produit connait alors un bug intermittent qui ne peut être corrigé que par un power reset de la carte. Après analyse, il se trouve que parfois les capteurs I2C ne répondent plus au microcontrôleur.

A votre avis, quel est le problème sur le bus I2C ?

L'application dépile des données d'un buffer FIFO (le capteur stocke les données dans sa mémoire interne dans un buffer, le microcontrôleur vient faire une requête des données a convenance. Cela permet d'éviter de perdre des données ou de devoir faire du pulling des données a la fréquence d'échantillonnage du capteur). Dans les données dépilées, si la trame de data correspond exactement à l'adresse du capteur voisin, avec le bit de read bien placé, et que l'octet suivant est une adresse de registre cohérent, les deux capteurs essaient de parler en même temps sur le bus ce qui leur cause une erreur du contrôleur de bus irrécupérable. Le seul moyen de corriger est de couper l'alimentation des capteurs pour réinitialiser le contrôleur de bus. Heureusement, la conception native de l'I2C protège le hardware (sorties open drain avec pull up).

Problème difficile a localiser car très peu fréquent. Le mode de détection est de placer un analyseur logique sur le bus, et de printer les logs de la carte. Toute ces données sont enregistrées en attente de la reproduction du BUG.

## UART

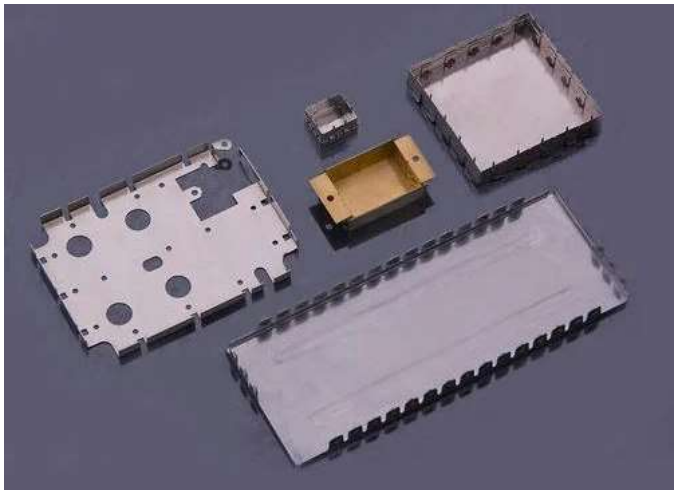
Sur un produit équipé d'un modem LTE/GNSS connecté au microcontrôleur avec un bus UART.
La configuration du modem fonctionne bien, l'accroche réseau est bonne et la transmission des données de l'application est bien en place. Le fonctionnement nominal de l'application est testé et validé.

Lors de la mise en place du FOTA (firmware over the air, mise à jour d'un produit à l'aide d'une technologie sans fil), les données du firmware de mise à jour sont systématiquement corrompues.

En fait lors du transfert des données de l'application, ces dernières transitent par la mémoire interne de modem et les transferts via le réseau sont déclenché par une commande AT. Il n'y a donc pas de transmission radio en même temps que les transmission UART.

En revanche le firmware étant trop grand pour être stocké dans la mémoire interne du modem, il est transmis au fil de la réception via l'UART. Le placement et les dimensions des pistes du bus UART en ont fait des antennes, qui captent donc les transmissions radio et ce qui cause des BITs flips et donc la corruption des données. Les bus de communications sont bien soumis aux bruits électromagnétiques, et la limite des protocoles dont les données ne sont pas contrôlées comme l'UART sont donc mises en évidence

- ⇨ En pratique, on essaie de placer les antennes et les bus à l'opposée de l'un et l'autre
- ⇨ Modifier la forme des pistes peut également jouer
- ⇨ Placer un shield emi sur la fonction logique, basé sur le principe de la cage de faraday



Pour valider que l'erreur venait bien des perturbation radio sur le bus, il a été possible de déporter l'Antenne LTE hors du produit (on coupe la piste radio et on soude un câble Radio blindé connecté a une antenne LTE. Déplacer l'antenne supprime complètement la corruption du firmware.

# 8. Le Direct Memory Access DMA

Le DMA, ou Direct Memory Access, est une technique utilisée dans les microcontrôleurs pour permettre à certains périphériques de transférer des données directement vers et depuis la mémoire sans solliciter le microcontrôleur lui-même.

Lorsqu'un périphérique, tel qu'une interface série, un convertisseur analogique-numérique (CAN) ou un module de capture d'événements, doit transférer une grande quantité de données vers ou depuis la mémoire du microcontrôleur, le DMA peut être utilisé. Il permet au périphérique de contourner le microcontrôleur et de communiquer directement avec la

mémoire, ce qui accélère les transferts de données et libère le microcontrôleur pour effectuer d'autres tâches.

Le fonctionnement du DMA dans un microcontrôleur est généralement le suivant :

1. Configuration : Le microcontrôleur est programmé pour configurer le contrôleur DMA en spécifiant les adresses de début et de fin de la zone mémoire où les données seront transférées, ainsi que le sens du transfert (lecture ou écriture).
2. Demande de transfert : Le périphérique demande l'accès au bus mémoire et au contrôleur DMA pour effectuer le transfert. Le microcontrôleur peut être notifié de cette demande via une interruption ou en vérifiant régulièrement l'état du contrôleur DMA.
3. Transfert direct : Une fois que le périphérique obtient l'accès au bus mémoire, il peut transférer les données directement vers ou depuis la mémoire sans passer par le microcontrôleur. Le contrôleur DMA se charge de l'acheminement des données et de la gestion des adresses.
4. Notification de fin : Une fois le transfert terminé, le contrôleur DMA peut notifier le microcontrôleur de sa complétion via une interruption ou d'autres mécanismes de signalisation.

Le DMA dans les microcontrôleurs offre des avantages similaires à ceux des ordinateurs, tels que :

- Amélioration des performances : En permettant aux périphériques d'accéder directement à la mémoire, le DMA réduit la charge sur le microcontrôleur et accélère les transferts de données, améliorant ainsi les performances globales du système.
- Libération du microcontrôleur : En évitant au microcontrôleur de gérer les transferts de données, le DMA libère sa capacité de traitement pour d'autres tâches, ce qui est particulièrement précieux dans les systèmes embarqués où les ressources sont limitées.
- Transferts asynchrones : Le DMA permet des transferts de données asynchrones, ce qui signifie que le périphérique et le microcontrôleur peuvent fonctionner de manière indépendante. Cela facilite la réalisation de tâches simultanées et peut améliorer la réactivité du système.

Il convient également de prendre en compte les défis et risques potentiels liés à l'utilisation du DMA dans les microcontrôleurs, tels que la gestion des accès concurrents à la mémoire et la nécessité de mettre en place des mécanismes de protection appropriés pour garantir l'intégrité des données.

En résumé, le DMA dans les microcontrôleurs est une technique qui permet à certains périphériques de transférer directement des données