Chapitre 3

Communication Série Synchrone Et Asynchrone

Ce chapitre, structuré sur trois semaines, constitue un module fondamental dédié aux protocoles de communication série. Ces protocoles représentent l'épine dorsale de toute interaction entre un microcontrôleur et ses périphériques externes, qu'il s'agisse de capteurs, d'actionneurs, de modules de communication ou de mémoires. La maîtrise de ces différents bus est essentielle pour tout concepteur de systèmes embarqués, car elle conditionne la capacité à intégrer des composants variés au sein d'une architecture cohérente et performante. Chaque protocole sera étudié selon une approche progressive, combinant l'analyse théorique des spécifications électriques et temporelles, la configuration pratique des registres associés sur le microcontrôleur cible, et la mise en œuvre d'exemples concrets d'interfaçage avec des composants du marché.

3.1. La Communication Asynchrone – UART/USART

La première semaine est consacrée à l'étude de la communication asynchrone, principalement représentée par les protocoles UART (Universal Asynchronous Receiver Transmitter) et USART (Universal Synchronous Asynchronous Receiver Transmitter), ce dernier offrant également une capacité synchrone optionnelle. Contrairement aux protocoles synchrones, la communication asynchrone ne nécessite pas de signal d'horloge distinct transmis entre les équipements ; la synchronisation est réalisée par accord préalable sur le rythme de transmission, appelé débit en bauds (baud rate). Ce débit, exprimé en bits par seconde, doit être rigoureusement identique entre l'émetteur et le récepteur pour garantir l'intégrité des données échangées.

3.1.1. Format de la trame UART

La transmission asynchrone s'organise en trames, dont la structure constitue un élément central du protocole. Une trame typique débute par un bit de start, qui marque le début de la transmission et permet au récepteur de se synchroniser. Suivent ensuite les bits de données, généralement configurés sur 7 ou 8 bits, qui transportent l'information utile. Un bit de parité peut être inséré optionnellement pour assurer une forme rudimentaire de détection d'erreurs ; il peut être configuré en parité paire, impaire ou omis. Enfin, un ou deux bits de stop viennent clore la trame, indiquant la fin de la transmission et laissant le temps au récepteur de se préparer pour la trame suivante. La compréhension de cette structure est fondamentale, car elle conditionne la configuration des registres du microcontrôleur et l'interprétation des données reçues.

3.1.2. Configuration pratique et applications

La mise en œuvre d'une liaison UART nécessite la configuration minutieuse de plusieurs paramètres : le débit en bauds, déterminé par la fréquence d'horloge du microcontrôleur et la valeur d'un registre de prédivision ; le format de la trame, via les bits de configuration du registre de contrôle ; et la gestion des interruptions pour une communication non bloquante. Les applications typiques couvertes durant cette semaine incluent l'interfaçage avec des capteurs série, tels que les GPS ou certains capteurs de distance, ainsi que l'intégration de modules Bluetooth comme les HC-05 ou HC-06, permettant d'établir des liens de communication sans fil avec des terminaux ou des applications mobiles. La mise en place d'un shell de débogage via UART, permettant d'interagir avec le système en cours de fonctionnement, sera également abordée comme outil de développement essentiel.

3.2 Bus Synchrone I2C – Inter-Integrated Circuit

La deuxième semaine se concentre sur le protocole I2C (Inter-Integrated Circuit), développé par Philips Semiconductor (aujourd'hui NXP). Il s'agit d'un bus synchrone multi-maîtres, fonctionnant sur seulement deux lignes : SDA (Serial Data Line) pour les données et SCL (Serial Clock Line) pour l'horloge de synchronisation. Cette frugalité en termes de nombre de broches en fait un protocole particulièrement apprécié pour interconnecter de nombreux périphériques sur une même carte électronique, d'autant que le bus supporte théoriquement jusqu'à 127 ou 1023 périphériques selon la version du protocole.

3.2.1 Architecture maître-esclave et adressage

Le protocole I2C repose sur une architecture maître-esclave où le maître initie et contrôle toutes les communications. Chaque esclave présent sur le bus possède une adresse unique sur 7 bits (ou 10 bits dans les extensions récentes), ce qui permet au maître de sélectionner le périphérique avec lequel il souhaite échanger avant chaque transaction. L'étude détaillée de la séquence de communication constitue un point central de cette semaine : la condition de start (démarrage) initiée par le maître en faisant descendre SDA alors que SCL est haut, suivie de l'envoi de l'adresse de l'esclave accompagnée du bit de lecture ou d'écriture, puis de l'accusé de réception (ACK) émis par l'esclave sélectionné, et enfin de la condition de stop qui libère le bus. Les temps morts, les conditions de répétition de start (restart), et la gestion des acquittements seront analysés en détail.

3.2.2 Utilisation avec capteurs et mémoires

La mise en œuvre pratique du bus I2C sur microcontrôleur nécessite la configuration des registres de contrôle pour définir la fréquence de l'horloge SCL (standard à 100 kHz, rapide à 400 kHz, ou haute vitesse jusqu'à 3,4 MHz), la gestion des interruptions pour détecter les événements de communication, et l'implémentation des routines de lecture et d'écriture respectant la temporisation imposée par le protocole. Les applications typiques abordées incluent l'interfaçage avec des capteurs environnementaux (température, humidité, pression) comme les BMP280, BME280 ou les séries HTU21D, qui intègrent souvent une interface I2C pour une communication simple et efficace. L'accès à des mémoires EEPROM externes via

I2C sera également étudié, permettant d'étendre les capacités de stockage non volatile du système embarqué au-delà de la mémoire interne du microcontrôleur.

3.3 Bus Synchrone SPI – Serial Peripheral Interface

La troisième semaine est dédiée au protocole SPI (Serial Peripheral Interface), développé par Motorola. Contrairement à I2C, SPI est un bus synchrone full-duplex, ce qui signifie qu'il permet d'émettre et de recevoir simultanément des données, offrant ainsi des débits beaucoup plus élevés, typiquement de l'ordre de plusieurs dizaines de MHz. Cette performance en fait le protocole privilégié pour les applications nécessitant des transferts de données rapides, comme l'affichage graphique, le stockage de masse, ou la communication avec des convertisseurs numériques-analogiques haute vitesse.

3.3.1 Architecture et fonctionnement full-duplex

Le bus SPI repose sur quatre signaux principaux : MOSI (Master Out Slave In) pour la transmission du maître vers l'esclave, MISO (Master In Slave Out) pour la transmission de l'esclave vers le maître, SCK (Serial Clock) pour l'horloge générée par le maître, et SS (Slave Select) ou CS (Chip Select) pour la sélection individuelle des esclaves. Contrairement à I2C, SPI ne possède pas de mécanisme d'adressage intégré ; la sélection d'un périphérique se fait par l'activation de sa ligne de sélection dédiée, ce qui implique qu'un nombre accru d'esclaves nécessite autant de lignes de sélection supplémentaires. Cette caractéristique est à la fois un avantage en termes de simplicité et un inconvénient en termes d'occupation des broches.

3.3.2 Modes de fonctionnement CPOL et CPHA

Un aspect fondamental du protocole SPI réside dans ses quatre modes de fonctionnement, définis par la combinaison de deux paramètres : CPOL (Clock POLarity) qui détermine le niveau de repos de l'horloge (haut ou bas), et CPHA (Clock PHAse) qui détermine sur quel front de l'horloge (montant ou descendant) les données sont échantillonnées. La compréhension de ces modes est cruciale, car un désaccord entre le maître et l'esclave sur la configuration CPOL/CPHA conduit à une corruption systématique des données échangées. Nous étudierons en détail chacun des quatre modes (0, 1, 2, 3) et leur configuration via les registres du microcontrôleur.

3.3.3 Applications et mise en œuvre

La mise en œuvre pratique de SPI abordera la configuration du diviseur d'horloge pour ajuster le débit de transmission, la gestion des lignes de sélection multiples (y compris l'utilisation de la broche SS en mode maître), et l'implémentation des routines de transfert full-duplex. Les applications typiques couvriront l'interfaçage avec des écrans graphiques, tels que les modules OLED ou TFT utilisant le contrôleur ILI9341, où le débit élevé de SPI est indispensable pour maintenir des taux de rafraîchissement acceptables. L'interface avec des cartes SD (Secure Digital) en mode SPI sera également étudiée, permettant d'intégrer un système de fichier pour la journalisation de données ou le stockage de fichiers de configuration. Enfin, nous aborderons l'interfaçage avec des convertisseurs CAN externes haute résolution, qui

exploitent souvent SPI pour obtenir les performances nécessaires à l'acquisition de signaux analogiques rapides et précis