# Chapitre 6

# Interfaces du microcontrôleur

## Introduction

Ce dernier chapitre explore les interfaces de communication et de stockage qui étendent les capacités du microcontrôleur au-delà de son cœur processeur et de ses périphériques internes de base. Si les chapitres précédents ont permis de maîtriser la gestion des entrées-sorties parallèles, des temporisations et des interruptions, il s'agit désormais d'aborder les mécanismes qui permettent au microcontrôleur d'interagir durablement avec son environnement en conservant des informations après une mise hors tension, et de communiquer avec d'autres systèmes intelligents. Ces interfaces sont au cœur de la plupart des applications embarquées modernes, qu'il s'agisse de sauvegarder des paramètres de configuration, d'échanger des données avec un capteur numérique, ou d'établir une liaison de diagnostic avec un ordinateur. L'approche reste fidèle à celle adoptée tout au long du module : chaque interface est étudiée à travers l'analyse de son fonctionnement théorique, la configuration minutieuse de ses registres d'état et de contrôle, et l'implémentation d'exemples d'applications pratiques permettant de maîtriser concrètement les échanges de données.

## 6.1. La mémoire EEPROM : stockage non volatile de données

La mémoire EEPROM (Electrically Erasable Programmable Read-Only Memory) constitue la première interface de stockage abordée dans ce chapitre. Contrairement à la mémoire Flash qui contient le programme exécutable, l'EEPROM est spécifiquement conçue pour sauvegarder des données utilisateur qui doivent persister après une mise hors tension du système, telles que des paramètres de calibrage, des seuils de configuration, ou des historiques de mesures. L'étude commence par la présentation de l'architecture interne de cette mémoire, en insistant sur sa nature non volatile et sur les spécificités de son cycle de vie (nombre limité d'écritures, temps d'accès asymétrique entre lecture et écriture). La configuration des registres d'état et de contrôle est ensuite détaillée : il s'agit de maîtriser les registres d'adresse pour pointer l'emplacement mémoire visé, le registre de données pour la valeur à lire ou à écrire, et surtout le registre de contrôle qui gère les phases délicates de l'écriture. Les procédures de lecture sont présentées comme relativement simples et rapides, tandis que l'écriture nécessite une séquence précise d'instructions pour éviter toute modification intempestive due à des aléas de fonctionnement (rebond d'alimentation, reset intempestif). Des exemples d'applications pratiques illustrent la sauvegarde et la restauration de paramètres, en intégrant des mécanismes de vérification comme la lecture après écriture ou l'utilisation de mots de validation pour garantir l'intégrité des données stockées.

### 6.2. Les liaisons série asynchrones : l'USART

La première interface de communication série abordée est l'USART (Universal Synchronous Asynchronous Receiver Transmitter), un périphérique particulièrement polyvalent qui supporte à la fois les communications synchrones et asynchrones. L'accent est mis principalement sur le mode asynchrone, largement utilisé pour établir des liaisons simples et robustes avec des équipements externes tels qu'un PC (via un convertisseur USB-UART), des modules GPS, des afficheurs, ou encore des modules Bluetooth. L'étude débute par la compréhension du format de trame série : bit de start, bits de données (généralement 8 bits), bit de parité optionnel, et bits de stop. La configuration des registres d'état et de contrôle est ensuite analysée en détail : les registres de configuration permettent de définir la vitesse de transmission (baud rate) à partir de l'horloge système, de choisir le format de la trame, et d'activer ou non l'interruption associée. Les registres d'état informent en temps réel de l'état du buffer de transmission (vide ou occupé) et de la réception (donnée disponible, erreur de trame, erreur de débordement). Enfin, les registres de données servent de point d'accès aux buffers de transmission et de réception. Les exemples d'applications pratiques couvrent l'envoi de chaînes de caractères vers un terminal série sur PC, la réception et l'interprétation de commandes simples, ainsi que la mise en place d'une communication bidirectionnelle avec un capteur externe. L'utilisation des interruptions pour la gestion des réceptions asynchrones est également abordée, permettant d'optimiser le temps processeur.

### 6.3. Le module MSSP : communication synchrone avec SPI et I2C

Le chapitre se poursuit par l'étude du module MSSP (Master Synchronous Serial Port), un composant matériel qui gère deux protocoles de communication synchrone majeurs dans le domaine des systèmes embarqués : le SPI (Serial Peripheral Interface) et l'I2C (Inter-Integrated Circuit). Ces deux protocoles sont omniprésents dans l'écosystème des capteurs modernes (accéléromètres, capteurs de température, écrans, mémoires externes) et leur maîtrise est essentielle pour tout développeur de systèmes embarqués.

#### *6.3.1. Le protocole SPI (Serial Peripheral Interface)*

Le protocole SPI est présenté comme une interface maître-esclave rapide, fonctionnant sur le principe de quatre fils principaux : l'horloge série (SCK), la sortie de données maître/entrée esclave (MOSI), l'entrée de données maître/sortie esclave (MISO), et les lignes de sélection d'esclave (SS). L'étude met en avant ses caractéristiques : communication full-duplex, vitesses de transfert élevées, et simplicité de mise en œuvre. La configuration du module MSSP en mode SPI est détaillée à travers l'analyse des registres de contrôle qui définissent le rôle maître ou esclave, la polarité et la phase de l'horloge (les quatre modes SPI), et la vitesse de transmission. Les registres d'état permettent de surveiller le déroulement des transferts et de détecter d'éventuelles collisions. Les exemples d'applications pratiques illustrent l'interface avec des composants typiques : lecture d'un convertisseur analogique-numérique externe, écriture dans une mémoire Flash série, ou encore pilotage d'un afficheur OLED. L'accent est

mis sur la gestion de la sélection des esclaves multiples et sur les séquences de transfert combinant envoi de commandes et réception de données.

### *6.3.2. Le protocole I2C (Inter-Integrated Circuit)*

Le protocole I2C est ensuite présenté comme une alternative particulièrement intéressante pour les bus de capteurs nécessitant une connectique réduite. Fonctionnant sur seulement deux fils (SDA pour les données, SCL pour l'horloge), il permet de connecter jusqu'à 127 esclaves sur un même bus grâce à un système d'adressage unique. L'étude met en lumière les particularités de ce protocole : communication half-duplex, mécanisme d'acquittement (ACK/NACK) pour valider chaque transfert, et gestion des collisions par arbitrage. La configuration du module MSSP en mode I2C est détaillée en distinguant le mode maître (le plus utilisé) du mode esclave. Les registres de contrôle permettent de définir la vitesse de communication (standard 100 kHz ou fast 400 kHz), d'activer les interruptions, et de générer les conditions de start et de stop. Les registres d'état jouent un rôle crucial pour suivre le déroulement des séquences complexes, en indiquant par exemple si un start a été émis, si une donnée a été transmise avec acquittement, ou si un conflit de bus est survenu. Les exemples d'applications pratiques couvrent les scénarios classiques : lecture de la température via un capteur I2C, configuration d'un expandeur d'entrées-sorties, ou communication avec une mémoire EEPROM externe (différente de l'EEPROM interne abordée en début de chapitre). Une attention particulière est portée à la construction des séquences de communication, qui impliquent l'envoi de l'adresse esclave suivie du registre cible avant les transferts de données proprement dits.

### Synthèse et applications transversales

La dernière partie de ce chapitre propose une synthèse comparative des différentes interfaces étudiées, mettant en évidence leurs domaines d'application privilégiés, leurs avantages et leurs limitations. L'EEPROM se distingue pour le stockage permanent de données en petite quantité ; l'USART est privilégiée pour les liaisons avec des équipements hétérogènes et le diagnostic ; le SPI est choisi pour les communications rapides avec des périphériques nécessitant un débit élevé ; l'I2C est idéal pour interconnecter de nombreux capteurs avec un câblage minimal. Des applications transversales sont proposées, combinant plusieurs interfaces au sein d'un même projet : par exemple, un système de télémétrie qui acquiert des données via un capteur I2C, les sauvegarde périodiquement en EEPROM, et les transmet à un PC via l'USART. Ces travaux pratiques de synthèse permettent aux étudiants de consolider leur compréhension des mécanismes de configuration des registres et d'acquérir une vision systémique de l'intégration des interfaces dans une application embarquée complète.