Chapitre 2

Architecture du microcontrôleur

Objectif du chapitre

L'objectif de ce chapitre est d'explorer en profondeur l'architecture interne et externe du microcontrôleur. Cette exploration se veut double : d'une part, une analyse matérielle qui détaille les composants physiques et leurs interactions ; d'autre part, une analyse logicielle qui décrit les mécanismes d'accès aux données et d'exécution des instructions. La maîtrise de ces deux aspects est fondamentale pour tout développeur de systèmes embarqués, car elle conditionne la compréhension du fonctionnement réel du composant et permet d'écrire un code à la fois efficace et parfaitement adapté aux ressources disponibles.

2.1. Architecture matérielle externe

L'architecture matérielle externe d'un microcontrôleur se réfère à l'ensemble des signaux accessibles physiquement par l'intermédiaire de ses broches de connexion. Chaque broche remplit une fonction spécifique, déterminée par le constructeur et parfois configurable par logiciel. Les catégories principales incluent les broches d'alimentation (Vdd, Vss) qui assurent la tension d'alimentation et la référence de masse, indispensables au fonctionnement du circuit. Les broches liées à l'oscillateur (OSC1, OSC2) permettent de connecter un résonateur ou un oscillateur externe pour générer le signal d'horloge qui cadence toutes les opérations internes. Les broches de reset (MCLR) offrent la possibilité de réinitialiser matériellement le microcontrôleur, le ramenant à un état connu en début d'exécution. Enfin, les broches d'entrées-sorties (E/S) à usage général constituent l'interface principale avec le monde extérieur : elles peuvent être configurées individuellement comme entrées pour lire des capteurs, comme sorties pour commander des actionneurs, ou encore comme supports de fonctions spécialisées (communication série, entrées analogiques, etc.). La compréhension du brochage et de l'affectation multiple des broches (multiplexage) est essentielle pour concevoir correctement le circuit électronique autour du microcontrôleur et éviter les conflits de signaux.

2.2. Architecture matérielle interne

L'architecture matérielle interne décrit l'organisation des différents blocs fonctionnels intégrés à l'intérieur de la puce du microcontrôleur, ainsi que les mécanismes de communication entre ces blocs. Au cœur de cette architecture se trouve le processeur (ou cœur CPU), qui exécute les instructions en effectuant des opérations arithmétiques et

logiques, en contrôlant le flux du programme et en gérant les transferts de données. Le processeur est accompagné de différents types de mémoires : la mémoire programme (généralement de type Flash ou ROM) qui stocke de manière non volatile le code à exécuter ; la mémoire de données vive (RAM) qui contient les variables temporaires, la pile et les structures dynamiques pendant l'exécution ; et parfois une mémoire EEPROM dédiée au stockage permanent de données utilisateur. L'ensemble de ces éléments communique via un système de bus internes, composés d'un bus d'adresses pour sélectionner l'emplacement mémoire ou le registre visé, d'un bus de données pour transporter les informations, et d'un bus de contrôle pour orchestrer les opérations (lecture, écriture, validation). Contrairement aux architectures à base de microprocesseurs où ces bus sont visibles sur les broches externes, dans un microcontrôleur ils sont internes, ce qui permet un fonctionnement plus compact, plus rapide et moins sensible aux perturbations électromagnétiques. L'architecture interne inclut également un ensemble de périphériques intégrés (timers, convertisseurs, interfaces de communication) connectés au bus, qui seront étudiés en détail dans les chapitres ultérieurs.

2.3. Architecture logicielle : modes d'adressage

L'architecture logicielle d'un microcontrôleur se manifeste d'abord à travers ses modes d'adressage, c'est-à-dire les différentes manières dont une instruction peut désigner l'emplacement des données sur lesquelles elle doit opérer. Le mode d'adressage immédiat consiste à fournir la valeur elle-même directement dans l'instruction, ce qui est rapide mais limité aux constantes. Le mode d'adressage direct spécifie l'adresse mémoire où se trouve la donnée, permettant d'accéder à des variables stockées en RAM ou à des registres spécialisés. Le mode d'adressage indirect utilise un registre qui contient l'adresse de la donnée, offrant ainsi une grande flexibilité pour parcourir des tableaux ou implémenter des structures de données complexes. D'autres modes existent selon les familles de microcontrôleurs, comme l'adressage relatif (ou indexé) qui combine une adresse de base et un décalage, particulièrement utile pour les accès aux éléments de tableaux ou aux champs de structures. Le choix du mode d'adressage approprié influence directement la taille du code généré, le nombre de cycles d'exécution nécessaires et, par conséquent, l'efficacité globale du programme. La maîtrise de ces modes est donc un prérequis indispensable pour une programmation bas niveau performante.

2.4. Architecture logicielle : jeu d'instructions

Le jeu d'instructions constitue le vocabulaire que le microcontrôleur est capable de comprendre et d'exécuter. Chaque instruction est représentée par un code binaire (opcode) et correspond à une opération élémentaire telle qu'un transfert de données entre registres, une opération arithmétique (addition, soustraction), une opération logique (ET, OU, XOR), une opération de décalage, une comparaison, ou encore une instruction de branchement conditionnel ou inconditionnel modifiant le déroulement du programme. La complexité du jeu d'instructions varie selon l'architecture du microcontrôleur : les architectures RISC (Reduced Instruction Set Computer) privilégient un nombre réduit d'instructions, chacune s'exécutant

généralement en un seul cycle d'horloge, ce qui facilite le pipelinage et la prévisibilité temporelle. À l'inverse, les architectures CISC (Complex Instruction Set Computer) proposent des instructions plus élaborées capables d'effectuer en une seule instruction des opérations qui nécessiteraient plusieurs instructions RISC, mais avec une durée d'exécution variable. Dans le cadre de ce cours, l'étude du jeu d'instructions se fera sur le microcontrôleur cible utilisé en travaux pratiques, en se concentrant sur les instructions les plus couramment utilisées. La connaissance approfondie de ce jeu permet non seulement de lire et d'écrire du code assembleur, mais aussi de comprendre le code généré par le compilateur C, d'identifier les inefficacités potentielles et d'optimiser les parties critiques des applications.

2.5. Liens entre architecture matérielle et logicielle

L'architecture matérielle et l'architecture logicielle ne peuvent être dissociées : elles sont les deux faces indissociables d'un même système. Les registres internes du processeur, comme le compteur programme (PC) qui pointe sur l'instruction en cours d'exécution, le registre d'état (Status) qui conserve les indicateurs de résultat (retenue, dépassement, résultat nul, etc.), ou le pointeur de pile (Stack Pointer) qui gère les appels de fonctions et les interruptions, sont des éléments matériels directement exploités par le jeu d'instructions. De même, l'organisation de la mémoire, avec ses différentes zones (Flash, RAM, registres de configuration des périphériques), est accessible via les modes d'adressage qui permettent au logiciel d'interagir avec le matériel. Cette interconnexion se manifeste concrètement lorsqu'un programmeur configure un périphérique : il écrit dans des registres de contrôle situés à des adresses spécifiques de l'espace mémoire, et ces écritures déclenchent des actions matérielles. Inversement, la lecture d'un registre de données donne accès à l'état matériel du système. La compréhension de cette symbiose entre matériel et logiciel est ce qui distingue fondamentalement la programmation des microcontrôleurs de la programmation sur système d'exploitation généraliste, et elle constitue le socle sur lequel reposent tous les développements embarqués qui seront abordés dans la suite du module.