Chapitre 1

Du microprocesseur au microcontrôleur

1.1 Définition d'un système microprogrammé

Un système microprogrammé est un système électronique numérique dont le comportement est défini par un programme exécuté par un processeur. Contrairement aux systèmes câblés dont la fonction est figée par la logique matérielle, un système microprogrammé offre une grande flexibilité puisqu'il suffit de modifier le programme pour en changer les fonctionnalités. Ces systèmes constituent le cœur de l'électronique moderne et se retrouvent dans des domaines aussi variés que l'électroménager, l'automobile, les télécommunications ou l'instrumentation médicale. Un système microprogrammé typique est composé d'un processeur, de mémoires (pour stocker le programme et les données), de périphériques d'entrée-sortie permettant l'interaction avec le monde extérieur, et d'un bus de communication assurant les échanges entre ces différents éléments. La maîtrise de l'architecture et de la programmation de ces systèmes est donc essentielle pour tout ingénieur œuvrant dans le domaine des systèmes embarqués.

1.2 Architectures de Von Neumann et de Harvard

L'architecture d'un processeur définit la manière dont il accède aux instructions et aux données. Deux modèles fondamentaux s'opposent : l'architecture de Von Neumann et l'architecture de Harvard. Dans l'architecture de Von Neumann, un seul bus est utilisé pour le transfert des instructions et des données, ce qui signifie que le processeur ne peut pas simultanément lire une instruction et accéder à une donnée en mémoire. Cette architecture, plus simple à implémenter, crée un goulot d'étranglement connu sous le nom de "bottleneck de Von Neumann", limitant les performances. À l'inverse, l'architecture de Harvard sépare physiquement les bus dédiés aux instructions et aux données, permettant au processeur de lire une instruction et une donnée simultanément. Cette séparation améliore considérablement le débit d'exécution. De nombreux microcontrôleurs modernes utilisent une variante appelée architecture de Harvard modifiée, qui conserve la séparation des bus tout en autorisant un certain accès du programme aux données en mémoire programme (par exemple pour les tables constantes).

1.3 Processeurs de types CISC et RISC

Les jeux d'instructions des processeurs se répartissent historiquement en deux grandes philosophies : CISC (Complex Instruction Set Computer) et RISC (Reduced Instruction Set Computer). L'approche CISC vise à fournir un jeu d'instructions riche et complexe, où chaque instruction peut effectuer des opérations élaborées (comme un accès mémoire suivi d'un calcul arithmétique). Cette complexité permettait à l'origine de réduire la taille du code,

mais au prix d'une architecture matérielle plus sophistiquée et d'un temps d'exécution variable selon les instructions. À l'opposé, l'approche RISC privilégie un jeu d'instructions réduit, où chaque instruction est simple, s'exécute généralement en un seul cycle d'horloge et opère principalement sur des registres. Les accès mémoire sont réalisés par des instructions spécifiques distinctes. Cette philosophie simplifie le matériel, facilite la mise en œuvre de techniques d'optimisation comme le pipeline, et permet d'atteindre des fréquences de fonctionnement plus élevées. La majorité des microcontrôleurs modernes, en particulier ceux destinés aux applications embarquées, adoptent une architecture de type RISC pour leur efficacité énergétique et leur performance prévisible.

1.4 Notion de pipeline

Le pipeline est une technique d'optimisation matérielle qui consiste à découper l'exécution d'une instruction en plusieurs étapes élémentaires (généralement : lecture de l'instruction, décodage, exécution, accès mémoire, réécriture du résultat), chacune étant traitée par un circuit dédié. Pendant qu'une instruction est en phase d'exécution, l'instruction suivante est décodée et la suivante est lue, créant ainsi un effet de chaîne de montage. Cette superposition des traitements permet d'atteindre un débit théorique d'une instruction par cycle d'horloge, améliorant considérablement le rendement du processeur. Cependant, le pipeline introduit des complexités, notamment en cas d'instructions de branchement (sauts conditionnels) qui peuvent vider le pipeline et nécessiter des mécanismes de prédiction pour maintenir l'efficacité. La profondeur du pipeline varie selon les architectures, et son efficacité dépend de la nature du code exécuté.

1.5 Microprocesseur ou microcontrôleur ?

La distinction entre microprocesseur et microcontrôleur est fondamentale pour le choix d'un composant. Un microprocesseur est un circuit intégré ne contenant que l'unité centrale de traitement (CPU). Pour former un système fonctionnel, il doit être associé à des composants externes : mémoire vive (RAM), mémoire morte (ROM ou Flash), et périphériques d'entrée-sortie. Cette architecture modulaire offre une grande flexibilité et des performances élevées, mais elle se traduit par une complexité de conception accrue, un encombrement plus important et une consommation souvent plus élevée. À l'inverse, un microcontrôleur intègre sur une même puce le processeur, la mémoire (RAM et Flash), et une multitude de périphériques (ports d'entrée-sortie, temporisateurs, convertisseurs analogique-numérique, interfaces de communication, etc.). Cette intégration en fait une solution autonome, compacte, économique et économe en énergie, idéale pour les applications embarquées où la taille, le coût et la consommation sont des contraintes critiques. Le choix entre ces deux solutions dépend donc des exigences spécifiques du projet : puissance et flexibilité pour le microprocesseur, intégration et simplicité pour le microcontrôleur.

1.6 Différentes familles de microcontrôleurs

Le marché des microcontrôleurs est extrêmement diversifié, avec de nombreuses familles répondant à des besoins variés. Parmi les architectures les plus répandues, on trouve les

microcontrôleurs 8 bits comme les AVR (utilisés dans les cartes Arduino), les PIC de Microchip, ou encore les STM8, qui offrent un excellent compromis entre simplicité, coût et consommation pour des applications modestes. Les microcontrôleurs 32 bits, principalement basés sur l'architecture ARM Cortex-M (familles STM32 de STMicroelectronics, Kinetis de NXP, etc.), dominent le marché des applications plus exigeantes en termes de puissance de calcul, de connectivité et de richesse en périphériques. D'autres familles comme les MSP430 de Texas Instruments se distinguent par leurs ultra-faibles consommations, idéales pour les applications alimentées par batterie. Chaque famille propose une gamme de composants déclinant différents niveaux de mémoire, de nombre de broches et de périphériques, permettant d'ajuster précisément le composant aux besoins de l'application.

1.7 Critères de choix du microcontrôleur

Le choix d'un microcontrôleur pour une application donnée repose sur une analyse multi-critères rigoureuse. Le premier critère est la puissance de calcul, déterminée par la taille du bus de données (8, 16 ou 32 bits) et la fréquence d'horloge. La capacité mémoire est également cruciale : la mémoire Flash pour le programme et la mémoire RAM pour les données doivent être suffisantes pour héberger le code et les variables, avec une marge pour les évolutions futures. La richesse des périphériques intégrés est un facteur déterminant : le nombre et les fonctionnalités des entrées-sorties, la présence de convertisseurs analogique-numérique (ADC), de temporisateurs (Timers), de modules de communication (USART, SPI, I2C, USB, CAN), etc. La consommation énergétique est primordiale pour les applications portables ou autonomes, avec des critères comme les modes de veille et le courant consommé en fonctionnement actif. Le coût unitaire, la disponibilité, la facilité de développement (existence d'outils de programmation, de cartes d'évaluation, de bibliothèques logicielles) et le support technique fourni par le fabricant sont également des considérations importantes qui influencent le cycle de développement et la pérennité du produit.

1.8 Adaptation au support de travaux pratiques

Afin d'assurer une cohérence entre les apports théoriques et les mises en pratique, le contenu de ce module s'appuie sur les cartes de développement disponibles dans les salles de travaux pratiques de l'établissement. Cette approche permet aux étudiants de manipuler concrètement un microcontrôleur tout au long du semestre, en confrontant directement les concepts abordés en cours à une réalisation pratique. Selon la plateforme utilisée (par exemple une carte Arduino basée sur un microcontrôleur AVR, une carte STM32 Nucleo, une carte PIC, ou tout autre support), les exemples de programmation, les configurations de registres et les applications présentées seront adaptés à ce composant cible. Cette orientation pragmatique garantit que les compétences acquises sont directement transférables aux séances de laboratoire et facilite l'appropriation des concepts par l'expérimentation.