Résumé: Cours 1

PINEL Johanne
OZBEKLER Erol
SUPLY Corentin
GARATE ANDEREGG Tomas

26 février 2024

1 Introduction

L'évolution de l'architecture de processeur ne cesse d'évoluer d'année en année. Nous passons graduellement d'une architecture CISC (Complex Instruction Set Computer) contenant un hardware compliqué accompagné d'instructions complexes qui produit des codes courts, à une architecture RISC (Reduced Instruction Set Computer) comportant beaucoup moins de transistors et des instructions simplifiées, mais générant des programmes plus longs. Cela est intéressant grâce à la baisse des prix des mémoires. La tendance de nos jours est l'ARM (Advanced RISC Machine), une famille d'architectures qui prend son essort à travers le monde entier.
Nous allons expliquer ce qu'est l'architecture ARM, en détaillant ses différentes familles, sa structure, son fonctionnement, en détaillant des exemples comme la clock et le timer qu'elle utilise. Nous aborderons également une manière de lire ce type d'architecture ainsi qu'une brève explication de ses diverses structures et de leur fonctionnement.

2 ARM : familles, structure, fonctionnement et clock

2.1 Familles

Il existe 3 différents types d'ARM. Les familles A, R et M. Chacune de ces familles est utilisée dans des domaines spécifiques. La famille A est principalement utilisée dans les smartphones et les appareils portables. La famille R est quant à elle adaptée aux solutions en temps réel, comme les contrôleurs, les équipements de réseau et les lecteurs multimédias. Finalement, la famille M, sur laquelle nous nous concentrerons, est elle destinée aux microcontrôleurs.
La famille M est divisée en plusieurs classes, chacune répondant à des besoins spécifiques en termes de coût et de performance (voir Fig. 1). La famille est composée de sous-catégories, de la classe M0, offrant une solution économique à faible consommation d'énergie pour des applications low-cost, à la classe M4, qui intègre des fonctionnalités avancées telles qu'une unité de calcul en virgule flottante (FPU) et des processeurs de signal numérique (DSP) pour le traitement de signal.

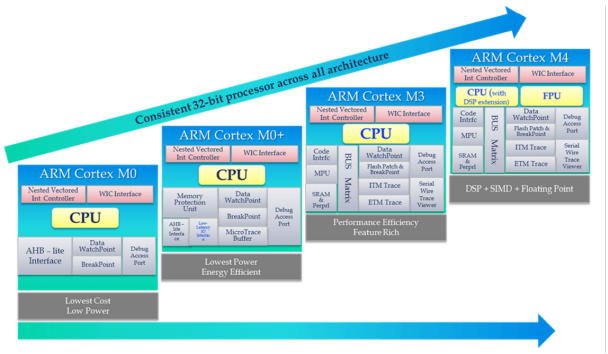

FIGURE 1 – Famille M, tiré du cours MICRO-315 du Prof. Francesco Mondada, slides 2024 p.29

L'entreprise ARM ne fabrique pas de processeurs ou de microcontrôleurs, elle vend une license à des entreprises qui mettent cette architecture dans leur microcontrôleur, comme dans notre cas le fait STMicroelectronics dans sa famille de microcontrôleurs STM32.

2.2 Structure

En figure 2 on peut voir à quoi ressemble une structure d'un microcontrôleur STM32F104 utilisé dans ce cours, basé sur un ARM Cortex M4.

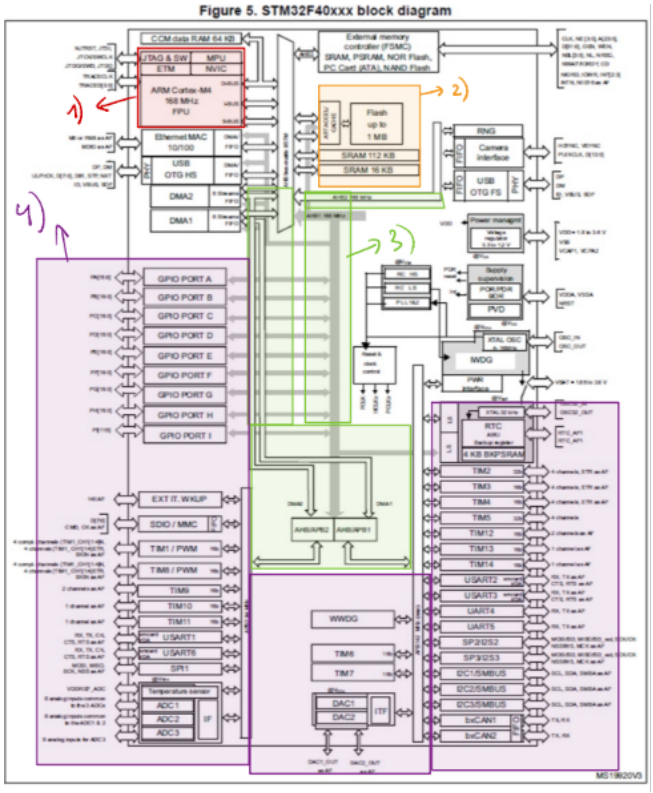

FIGURE 2 – Structure du STM32F104 utilisant ARM Cortex M4, tiré du datasheet du STM32F104.

Pour comprendre ce type de schéma, nous pouvons établir une méthodologie qui nous permettra de l'analyser efficacement. Celle-ci est la suivante :

1. Avoir un point de référence : le coeur du microcontrôleur (ARM cortex M4) (rouge)
2. Repérer où se situe la mémoire (Flash/RAM/SRAM) (orange)
3. Repérer les bus de communications (vert)
4. Repérer les périphériques qui sont liés aux bus. (violet)

2.3 Fonctionnement

Le cortex M4 est le coeur du microcontrôleur. La partie principale est naturellement le CPU, qui effectue les calculs et opérations nécessaires au fonctionnement du système, avec des instructions DSP pour le traitement de signal ainsi qu'un floating point unit (FPU) pour gérer les nombres à virgules. Le coeur possède aussi une matrice de bus qui permet de communiquer avec l'ensemble des modules qui se trouvent dans le cortex.
Autour du coeur se trouve la mémoire (SRAM/Flash/RAM) que le microcontrôleur utilise.
Pour interagir avec les périphériques, le système utilise deux bus et un registre d'accès direct à la mémoire (DMA) qui permet une communication directe avec la mémoire
Finalement, les pins de sortie des ports GPIO (acronyme pour General Purpose Input/Output) peuvent fonctionner soit en entrée, soit en sortie. C'est sur ces pins que les périphériques seront

connectés. Chaque pin peut prendre une fonction spécifique définie en fonction des besoins de l'application. La figure 3 illustre le schéma d'un pin I/O.

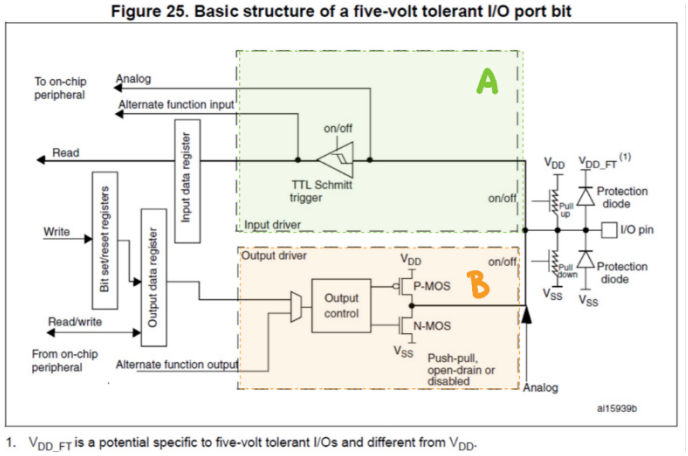

FIGURE 3 – Schéma d'un pin I/O, tiré du cours MICRO-315 du Prof. Francesco Mondada, slides 2024 p.39

Chaque pin possède des diodes de protection et des résistances programmables. Si le pin est en entrée alors le bloc A est sollicité. Dans le cas contraire, si le pin est en sortie alors ce sera le bloc B qui sera sollicité. Dans le cas où le pin fonctionne en entrée, on pourra alors choisir entre une entrée analogique ou digitale. Pour ce qui est de la sortie, on retrouve une structure en CMOS qui permet de générer la sortie.
Afin de choisir le mode I/O, on utilise 4 registres :

1. Mode (MODER) : si IN ou OUT .
2. Type d'output (OTYPER), où on peut avoir deux configurations de la structure CMOS. La première est une configuration push-pull et la deuxième configuration de type open-drain où on a soit une sortie au GND soit une sortie flottante.
3. Type de pull (PUPDR) : permet de piloter les deux résistances programmables.
4. Vitesse de sortie (OSPEEDR) : gère la vitesse de sortie.

2.4 Clock et Timer

Comprendre l'architecture d'une clock n'est pas une tâche facile. Une fois la clock (par exemple celle qui arrive au timer) identifiée sur le schéma, nous pouvons entamer une analyse. L'analyse est effectuée uniquement sur la partie qui nous intéresse.
Le timer remplit deux fonctions principales : il est utilisé pour générer des signaux réguliers (par exemple PWM) ou capturer des signaux et générer un timing précis. Le compteur (rouge sur la figure 4) du timer (CNT) est facilement identifiable car il a une clock en entrée (le block qui génère la clock est en orange sur le schéma). De plus, le timer est doté de registres qui peuvent être comparés au compteur, ainsi que capturer l'état du compteur sur la base d'un signal externe.
Il existe différentes structures de timer, mais ils possèdent tous les mêmes éléments de références (counter/clock/registres).

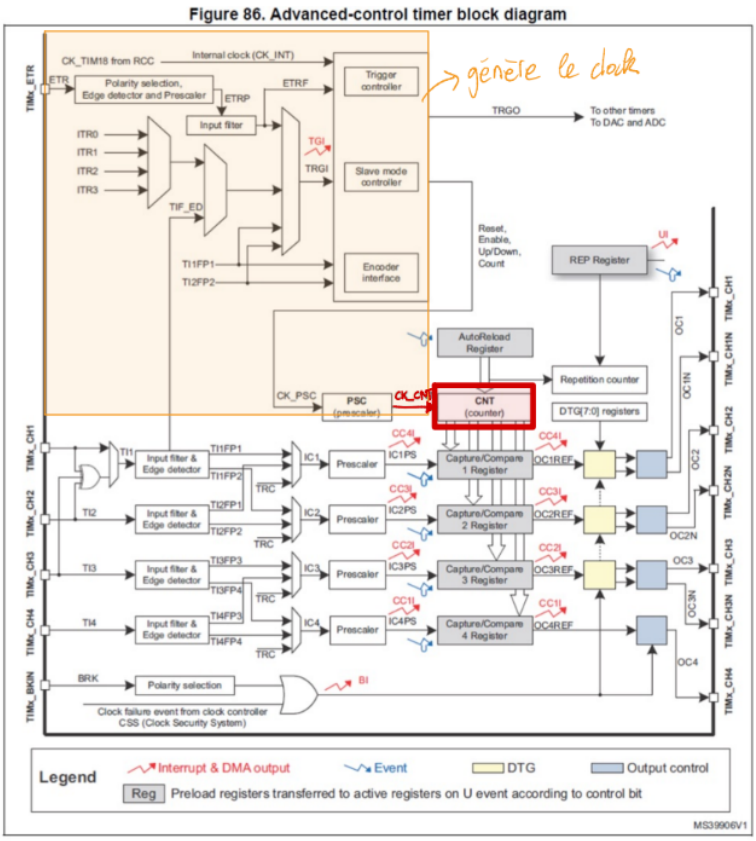

FIGURE 4 – Timer du STM, tiré du cours MICRO-315 du Prof. Francesco Mondada, slides 2024 p.52

Les ARM disposent d'un vaste ensemble d'instructions essentielles pour une architecture RISC, qui sont exécutées en général en un seul cycle d'horloge. Le jeu d'instructions s'étend lors de l'utilisation de processeurs Cortex plus avancés, par exemple par l'ajout des instructions DSP.
En programmation assembleur, on retrouve 16 registres, parmi lesquels 3 sont particuliers (SP/LR/PC), ainsi que des registres d'état. Les instructions sont généralement exécutées en un cycle d'horloge du processeur, à l'exception de la division qui est itérative et peut nécessiter jusqu'à 12 cycles. De plus, des instructions spécifiques sont disponibles pour le multitâche ou le traitement du signal, comme le DSP.