

ARCHITECTURE ARM

1. Introduction

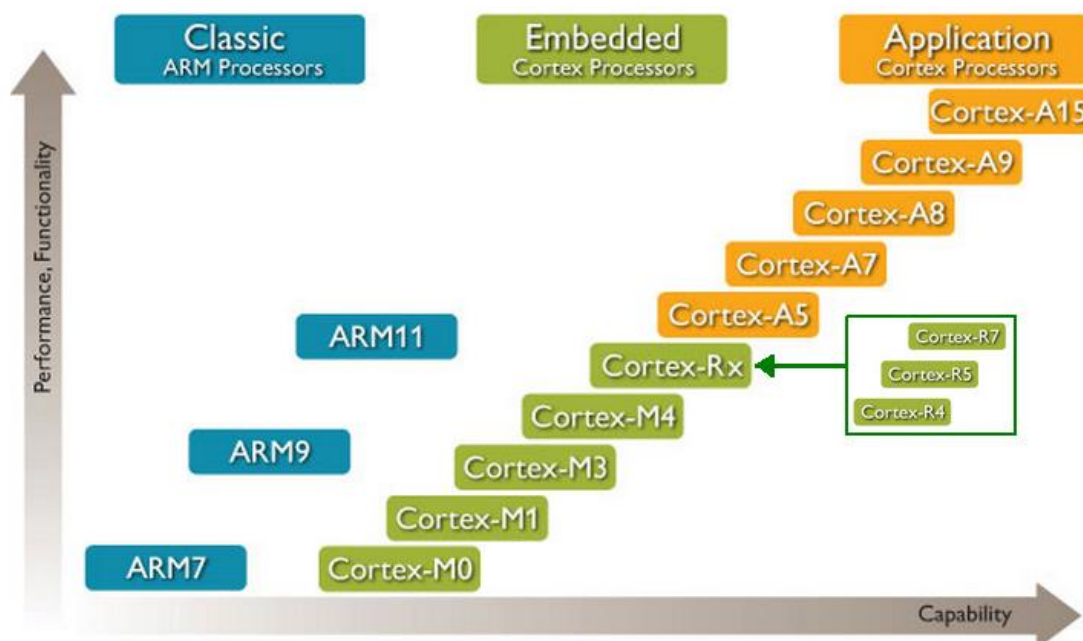
- ARM (Advanced RIS Machines) est une famille d'architectures de processeurs basées sur l'architecture RISC.
- Bien connu pour son efficacité énergétique; d'où largement utilisé dans les appareils mobiles, tels que les téléphones intelligents et les tablettes.
- L'entreprise fait la conception des processeurs ARM; ne fabrique pas, mais vend la licence à des partenaires de semi-conducteurs qui ajoutent leur propre propriété intellectuelle (IP) à la propriété intellectuelle d'ARM.

L'architecture cortex est divisée en trois familles :

CORTEX A (APPLICATION) : Microprocesseurs à haute performances, capables de supporter des systèmes d'exploitation. (Application : smartphones, digital TV, smart-books...)

CORTEX R (REAL TIME) : Hautes performances pour les applications temps réel, grande fiabilité. (Applications : Système de freinage automobile, les groupes motopulseurs, etc...)

CORTEX M (MICROCONTROLEUR) : Des solutions à faible coût pour des applications embarquées. (Applications : sonorisation, électroniques de carrosserie automobile et les airbags).



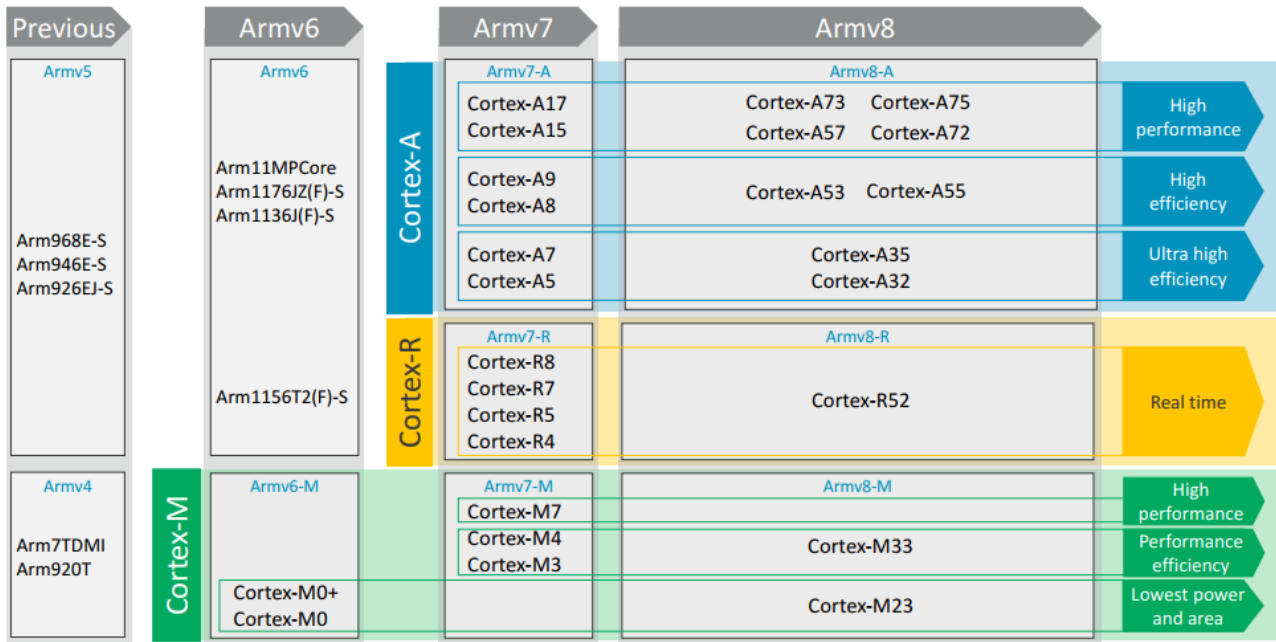
PROCESSEUR ARM VS ARCHITECTURE ARM

- **Architecture ARM**

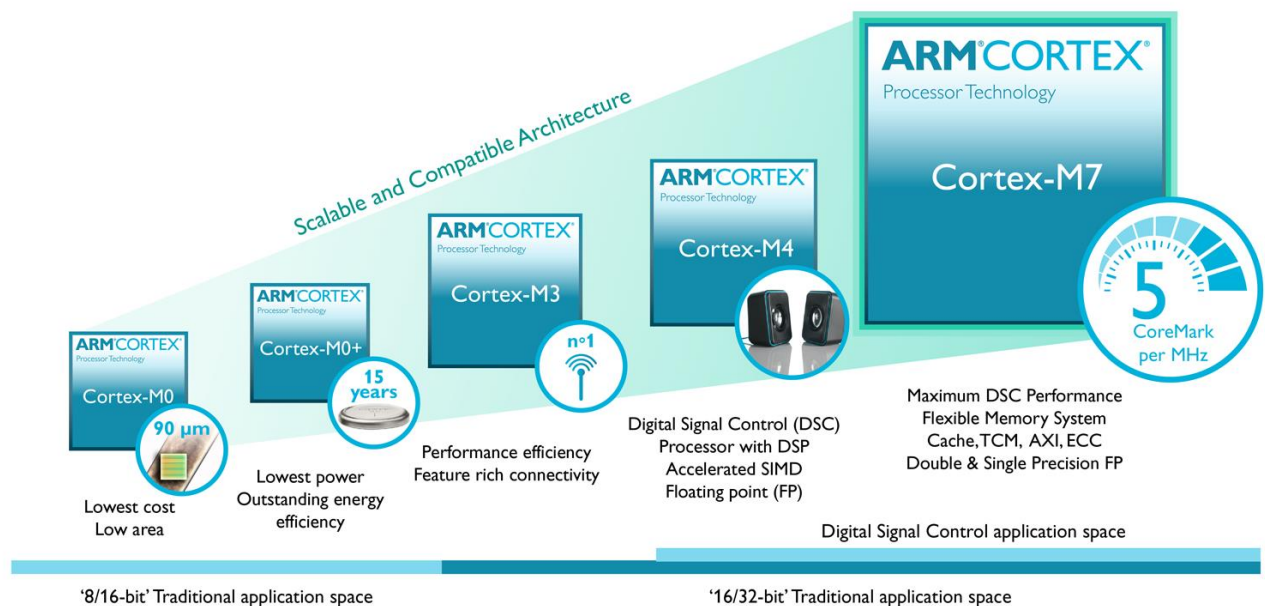
- Décrit les détails du jeu d'instructions, du modèle du programmeur, le modèle d'exception et le mappage de la carte mémoire.
- Documenté dans le Manuel de Référence de l'Architecture.

- **Processeur ARM**

- Développé en utilisant l'une des architectures ARM.
- Plus de détails de mise en œuvre.



2. Architecture Cortex M



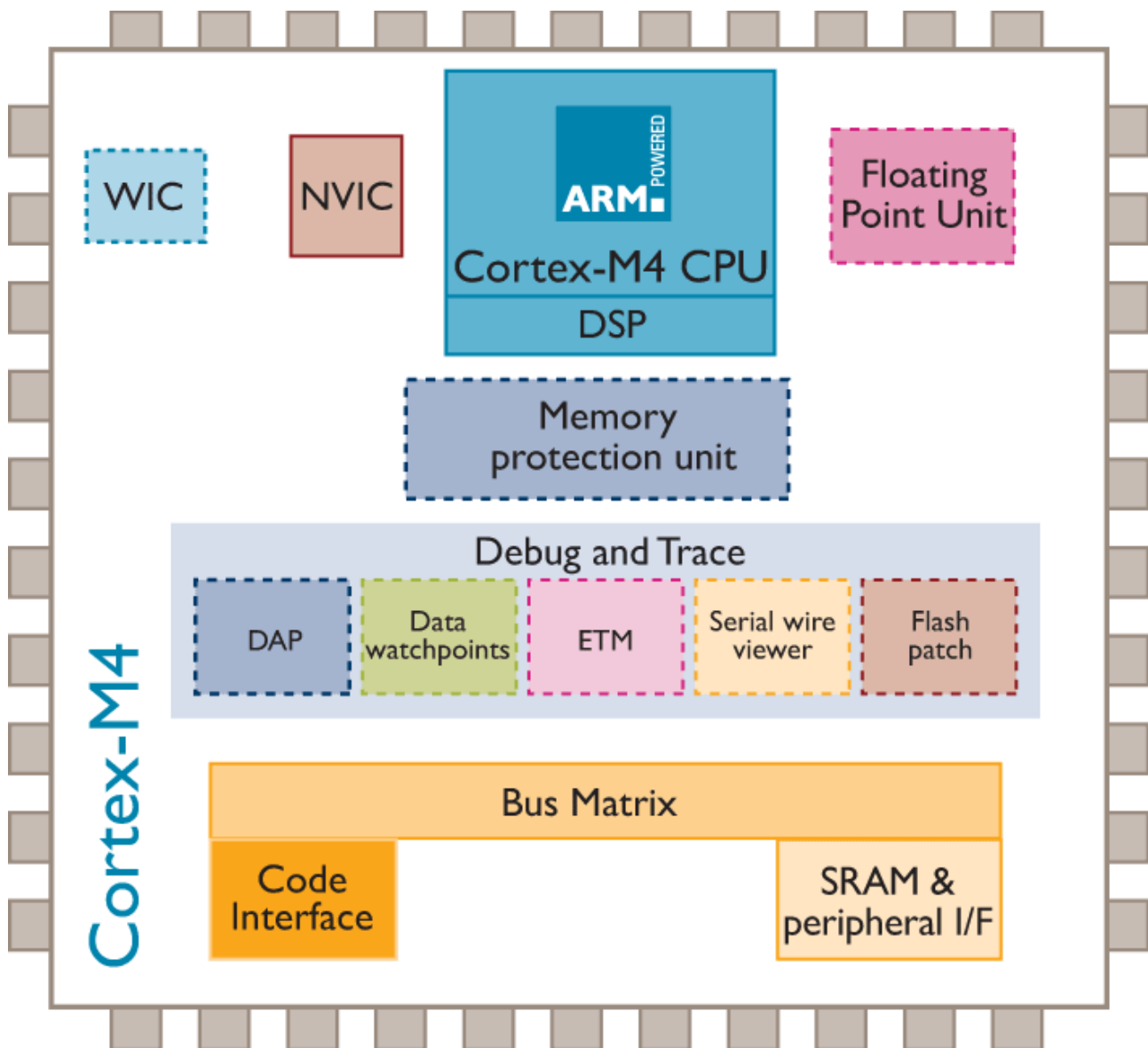
SIDM : Single Instruction Multiple Data

TCM : (Mémoire à Couplage Etroit) : ITCM (64 bits), DTCM (2 x 32 bits).

AXI : Advanced External Interface : connexion des mémoires externes avec un bus de 64 bits.

ECC : Error Code Correction.

Cortex M4



WIC : Wakeup Interrupt Controller (optionnel)

NVIC : Nested Vectored Interrupt Controller

ETM : Embedded Trace Macrocell (optionnel)

DAP : Debug Access Port (optionnel)

Cortex-M3 et Cortex-M4 sont des microcontrôleurs 32bits basés sur une architecture Armv7-M. Cortex-M4 a été étendu pour implémenter le profil d'une architecture Armv7E-M (FPU, Instructions DSP ...).

Cortex M possède un jeu d'instructions RISC. Toutes les opérations sur les données sont effectuées sur les registres du CPU, ce qui limite l'accès à la mémoire uniquement pour chargement et le sauvegarde des données.

Historiquement les processeurs ARM, possèdent un jeu d'instructions de 32 bits. Dans le bit d'optimisation de la mémoire, ARM a introduit les instructions Thumb sur 16bits. Par la suite, ARM a introduit le jeu d'instructions Thumb-2, qui est un mélange de jeux d'instructions de 16 et 32 bits.

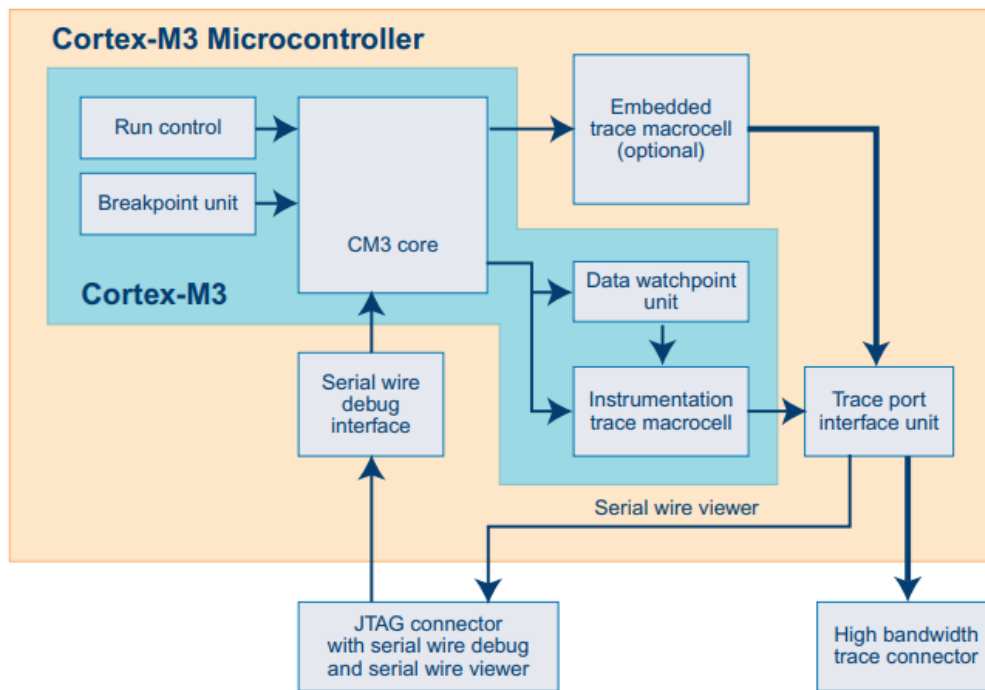
Cortex M3/M4/M7 sont conçus pour supporter les instructions Thumb et Thumb-2, avec les instructions de l'unité à virgule flottante et les instructions SIMD : *Single Instruction Multiple Data*.

S'appuyant sur une architecture **Hardvard**, le microcontrôleur Cortex M possèdent plusieurs bus, ce qui lui permet d'effectuer plusieurs opérations en parallèles. L'accès aux données peut être aligné sur 32 bits ou non aligné ; ces deux modes d'accès offrent une grande flexibilité et une bonne exploitation de la mémoire.

Grace un pipeline à trois niveaux les microcontrôleurs Cortex M exécutent la plupart des instructions en un cycle machine.

cortex-M3/M4 intègrent :

- Un contrôleur d'interruptions NVIC (Nested Vectored Interrupt Controller) déterministe et rapide, dont la latence est limité à 12 cycles.
- Un SysTick Timer 24 bits à auto-rechargement. Ce SysTick Timer fournit une interruption périodique pour l'implémentation d'un système d'exploitation temps réel (RTOS).
- Une architecture de débogage **CoreSight** au sein des processeurs Cortex-M. Il permet de placer jusqu'à huit points d'arrêt matériels dans des régions de code ou de données. CoreSight fournit également trois unités de trace distinctes qui prennent en charge des fonctionnalités de débogage avancées sans empiéter sur l'exécution du processeur.



Le Cortex-M3 et le Cortex-M4 sont toujours équipés d'une unité de Watchpoint et de trace de données (DWT) et d'une unité « Instrumentation trace macrocell » (ITM). L'interface de débogage permet à un débogueur peu coûteux d'afficher le contenu de la mémoire et des registres de périphériques «à la volée» sans arrêter le processeur. La deuxième unité de trace s'appelle « Trace port interface unit ». Cette unité de trace fournit une méthode de communication de débogage entre le code en cours d'exécution et l'interface utilisateur du débogueur. La troisième unité (optionnelle) de trace est appelée « embedded trace macrocell » (ETM). Cette unité est implémentée dans les microcontrôleurs haut de gamme afin de fournir des capacités de débogage étendues. L'ETM fournit des informations de trace d'instruction qui permettent au débogueur de créer un assembleur et une liste de trace du langage de haut niveau. L'ETM permet également d'utiliser des outils plus avancés tels que la couverture de code et l'analyse des performances de synchronisation.

MODES ET ÉTATS DE FONCTIONNEMENT :

Cortex M3/M4 dispose de deux états et des modes de fonctionnement. En plus le processeur peut avoir deux niveaux d'accès : privilégié et non privilégié. Le niveau privilégié permet l'accès à toutes les ressources du processeur, tandis que le niveau non privilégié n'autorise pas l'accès à quelques zones mémoires et interdit l'utilisation de quelques instructions.

Etats de fonctionnement

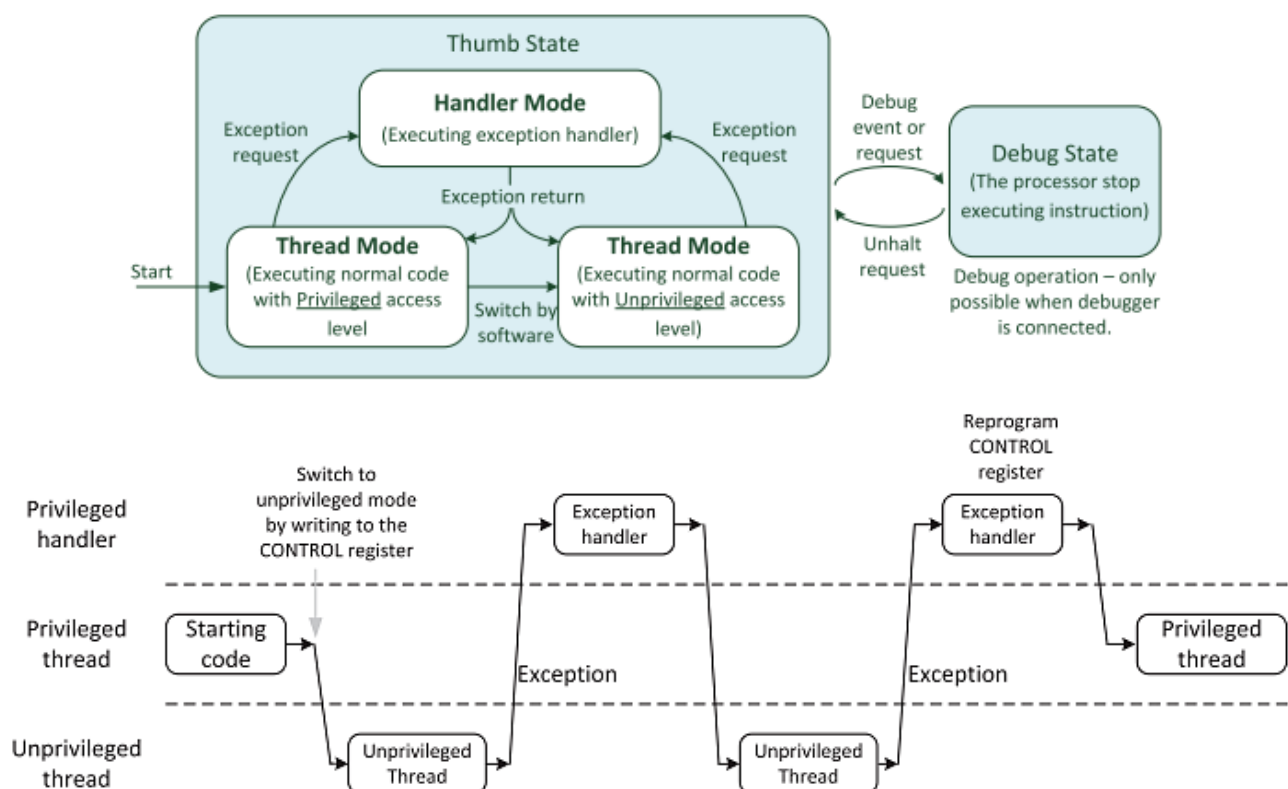
Etat de débogage : le processeur arrête l'exécution des instructions et passe à l'état de débogage.

Etat Thumb : le processeur exécute le code du programme.

Modes de fonctionnement

Handler mode : lors de l'exécution d'une routine d'interruption (ISR). Dans ce mode le processeur a toujours un accès privilégié.

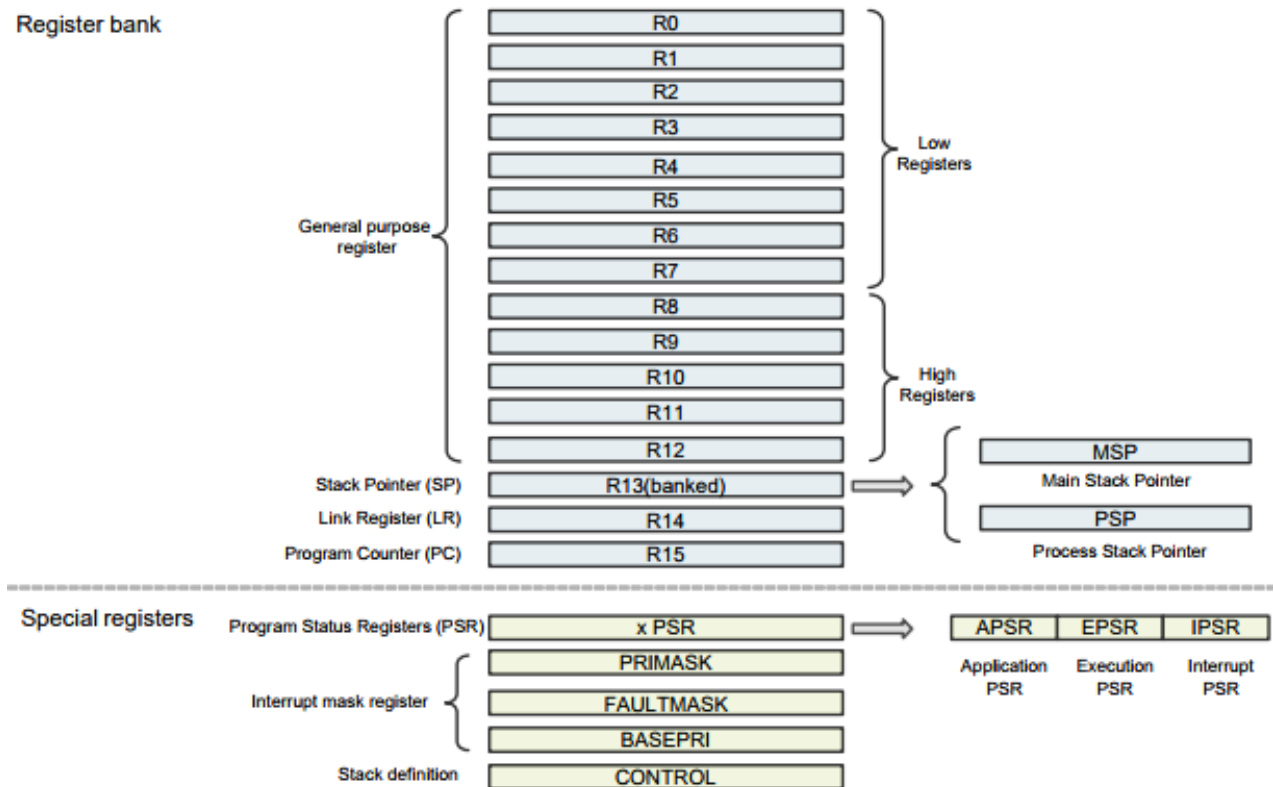
Thread mode : lors de l'exécution du code de programme normal. Dans ce mode le processeur peut être au niveau privilégié ou non privilégié. Dans des applications simples (sans système d'exploitation) le mode thread non privilégié est inutilisé.



2.1. Les Registres

Similaire à la plupart des microprocesseurs, l'architecture Cortex-M dispose d'un certain nombre de registres à l'intérieur du core du processeur pour le traitement et le contrôle des données. Cortex-M est une architecture de type **Load-Store** ; cela signifie s'il y a des données en mémoire à traiter, il doit transférer les données de la mémoire vers les registres, les traiter à l'intérieur du processeur, puis faire le transfert vers la mémoire si nécessaire. Ayant un nombre suffisant de registres dans la banque de registres, cet arrangement est facile à utiliser, et permet au compilateur C de générer un code de programme efficace.

La banque de registre cortex-M4 comporte 16 registres. Dont treize registres à usage général.



Les Registres Bas (R0 à R7) sont des registres à usage général accessibles par toutes les instructions, alors que les registres R8 à R12 sont accessibles par les instructions codées sur 32 bits, et un nombre limité d'instructions codées sur 16 bits.

R13 est un pointeur de pile (Stack Pointer). Physiquement il y a deux pointeurs de pile. MSP ou SP_Main et ou PSP SP_Process. MSP est le pointeur par défaut, il est utilisé en mode Handler (Interruption ou système d'exploitation). En dehors du mode handler, il est possible d'utiliser SP_Process.

R14 appelé aussi (LR : Link Register), chargé par l'adresse de retour lors de l'appel d'une fonction ou d'une procédure. Lorsqu'elle fonction veut appeler une autre fonction. Elle doit sauvegarder le contenu du registre LR dans la pile.

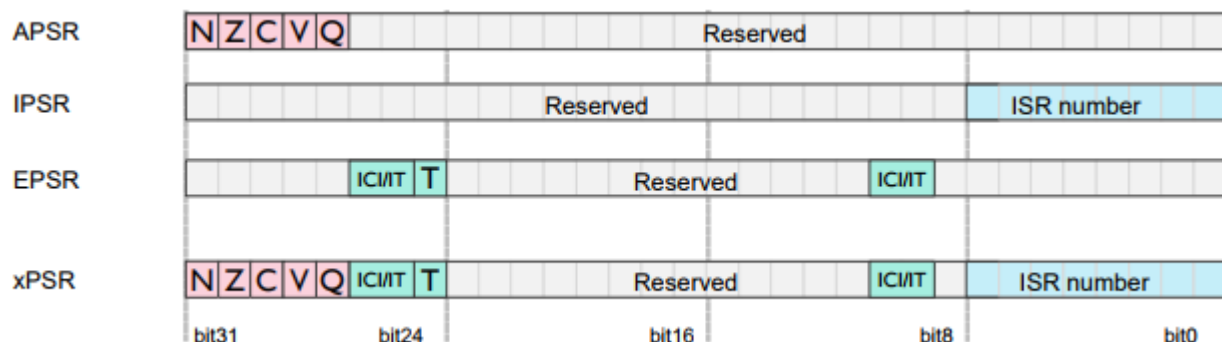
R15 joue le rôle de compteur de programme.

A part ces registres, il y a d'autres registres spéciaux; on peut citer ici le registre **xPSR**. Ce registre est composé de trois registres (APSR : Application Program Status Register ; EPSR : Execution Program Status Register ; IPSP : Interrupt Program Status Register).

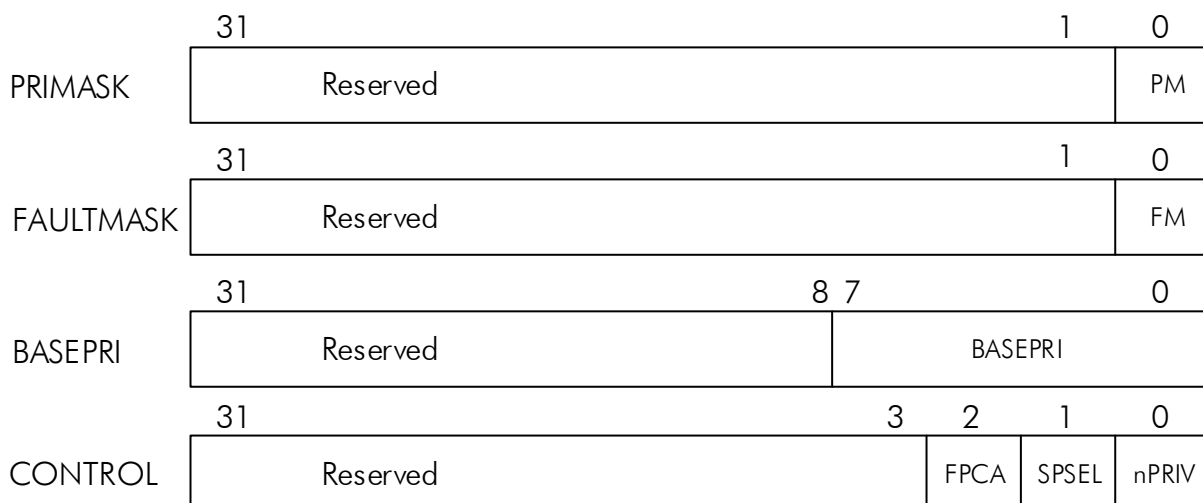
APSR : contient les indicateurs : de retenue C, de signe N, du résultat nul Z...

EPSR : indique si le microprocesseur exécute une instruction If-Then ou une instruction Load/Store Multiple Register.

IPSR : contient le numéro d'exception ou d'interruption.



Registres PRIMASK, FAULTMASK, BASEPRI et CONTROL



Registre PRIMASK :

lorsque le bit PM = 1, toutes les interruptions sont masquées, sauf NMI (Non-Maskable Interrupt) et l'exception HardFault.

Registre FAULTMASK :

Lorsque le bit FM = 1, l'exception HardFault est masquée.

Registre BASEPRI :

Fixe le niveau de priorité (voir chp. NVIC).

Registre CONTROL :

Bit FPCA : Lorsque le paramètre flottant est implémenté, ce bit indique si le contexte à virgule flottante est actuellement actif.

Bit SPSEL : définit le pointeur de pile actif (0 : MSP, 1 : PSP).

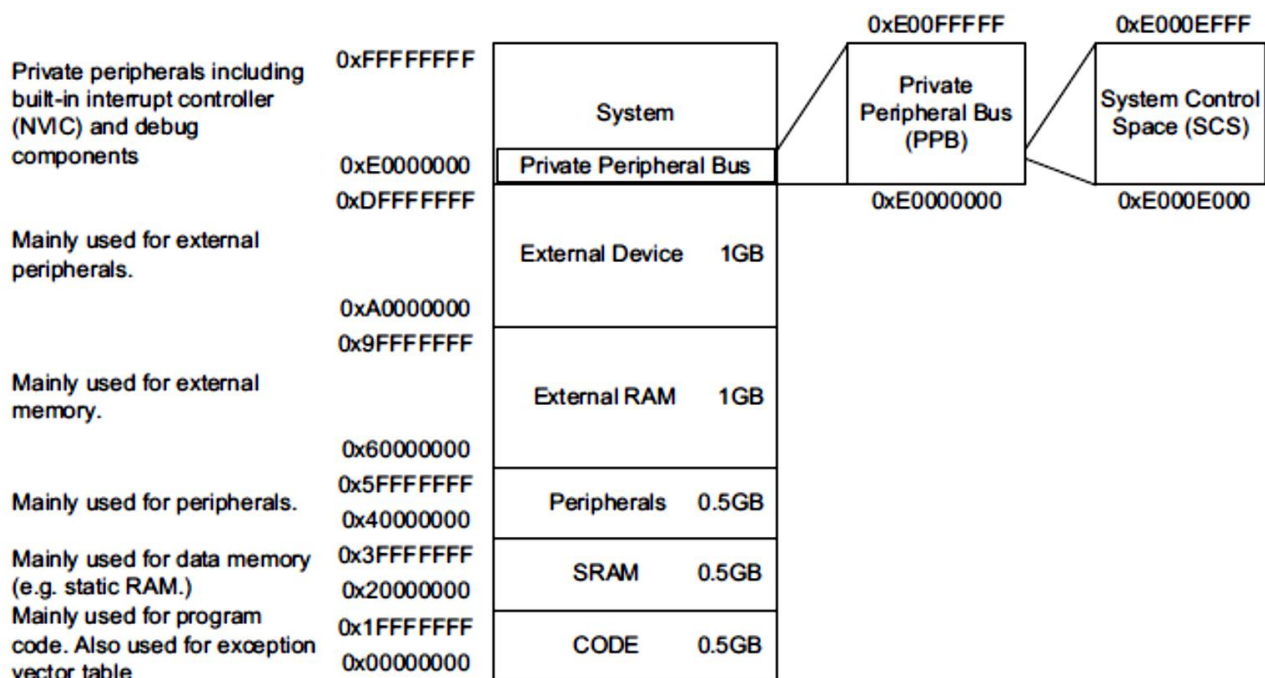
Bit nPRIV : définit le niveau d'exécution des Thread (0 : mode privilégié, 1 : non privilégié).

2.2. Mappage mémoire

ARM définit un espace d'adressage mémoire normalisé commun à tous les cœurs Cortex-M, ce qui garantit la portabilité du code entre différents fabricants de silicium. L'espace d'adressage a une largeur de 4 Go et divisé en plusieurs régions. Le partitionnement est basé sur des usages typiques de sorte que différentes zones sont conçus pour être utilisés principalement pour :

- Accès au code du programme (région de CODE)
- Accès de données (région de SRAM)
- Périphériques (région périphérique)
- contrôle et déboguer les composants internes du processeur (par exemple, le bus PPB)

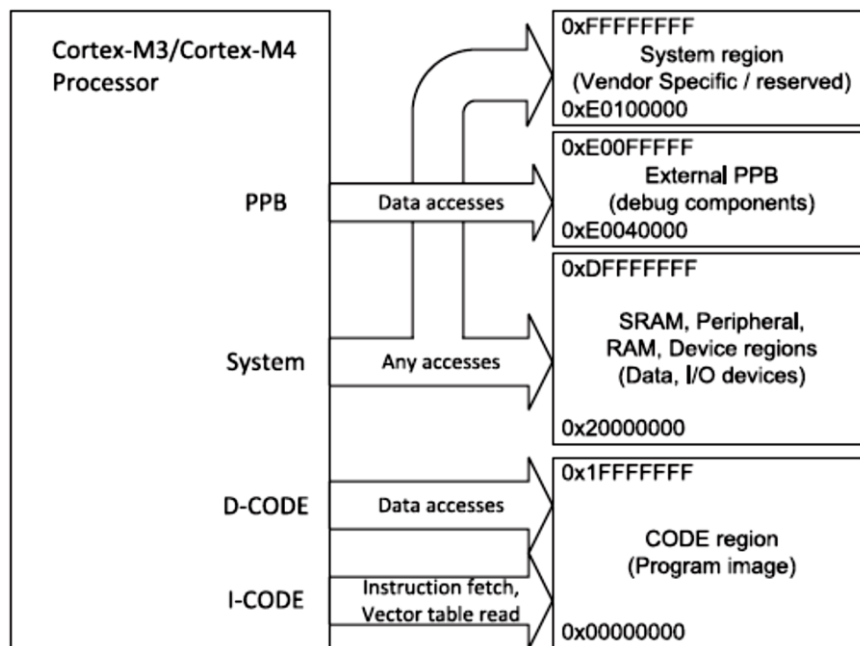
L'architecture permet également une grande flexibilité pour permettre à des régions de mémoire d'être utilisés pour d'autres fins. Par exemple, les programmes peuvent être exécutés à partir du CODE, ainsi que la région de SRAM, et un microcontrôleur peut également intégrer des blocs SRAM dans la région de CODE.



Dans la pratique, de nombreux microcontrôleurs utilisent seulement une petite partie de chaque région pour le programme flash, SRAM, et les périphériques. Certaines régions ne sont pas utilisées.

Le processeur contient quatre bus externes basés sur le protocole AMBA (Advanced Microcontroller Bus Architecture).

- Le bus ICode : principalement pour la mémoire de programme. Ce bus de 32 bits permet la recherche des instructions et des vecteurs à partir de l'adresse 0x00000000 à 0x1FFFFFFF. Basé sur le protocole AMBA 3.0 AHB Lite (AHB : Advanced Hi-performance Bus).
- Le bus Dcode : principalement pour la mémoire de programme. Il permet l'accès aux données à partir de l'adresse 0x00000000 à 0x1FFFFFFF. Basé sur le protocole AMBA 3.0 AHB Lite.
- Le bus système : principalement pour la RAM et les périphériques. Il permet l'accès à la totalité de l'espace compris entre 0x20000000 et 0xFFFFFFFF à l'exception de la région PPB. Basé sur le protocole AMBA 3.0 AHB Lite.
- Le bus PPB pour Private Peripheral Bus. Bus de débogage des composants privés au niveau d'adresse 0xE0040000 à 0xE00FFFFF. Basé sur le protocole AMBA 3.0 APB (Advanced Peripheral Bus) moins performant que l'AHB Lite.

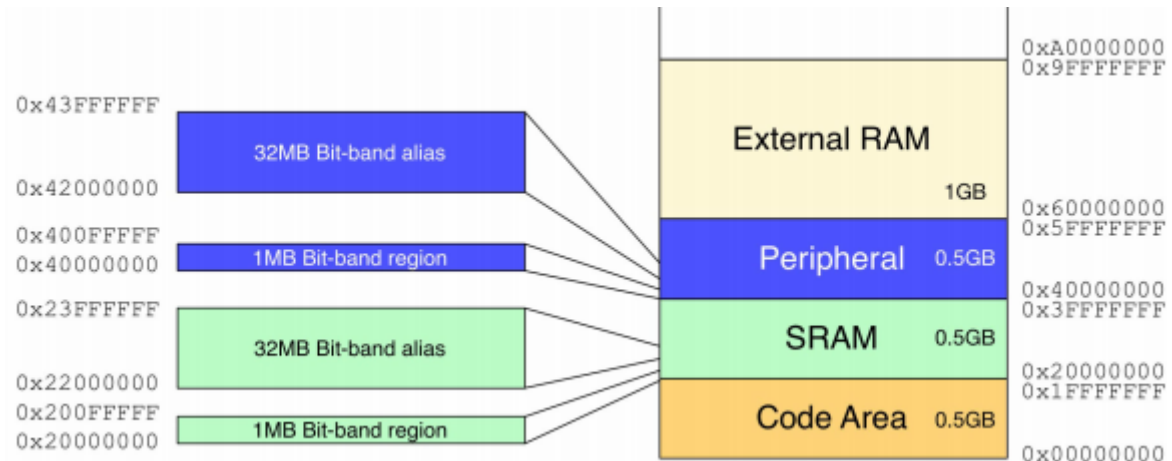


Les périphériques d'interface sont généralement basés sur le protocole APB. Cependant, le protocole AHB peut être utilisé pour les interfaces de haute performance.

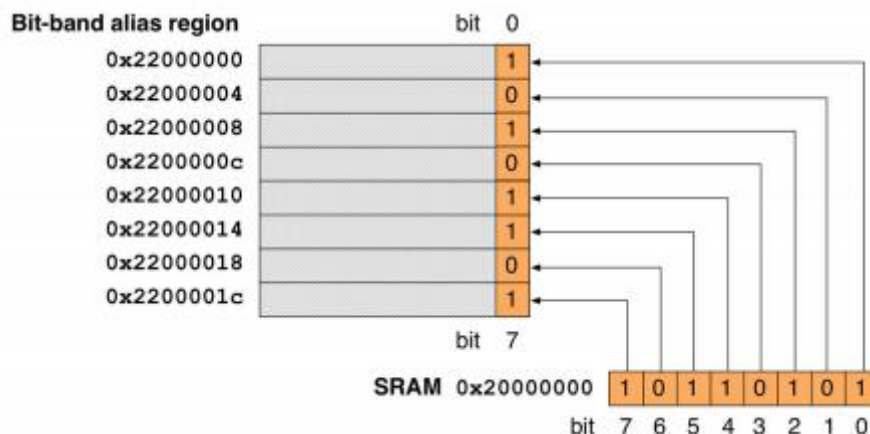
2.3. L'accès aux bits (bit banding)

Dans les applications embarquées, il est souvent nécessaire de lire et de changer l'état d'un bit de façon individuelle. Dans la plupart des cas, cet accès est effectué par masquage (AND et OR), qui nécessite une opération de lecture-modification-écriture. Bien que cela fonctionne bien, les processeurs cortex M fournissent une méthode plus efficace, dite *bit-banding*.

La technique utilisée, consiste à réserver à chaque bit toute une case mémoire alias.



Dans les processeurs cortex M, le premier mégaoctets (1Mo) de la zone SRAM et le premier mégaoctets (1Mo) de la zone des périphériques sont adressable par bit (region bit-banding). A chaque région bit-banding, une région de 32Mo alias lui est réservée. La manipulation d'un bit s'effectue en se référant à une adresse mémoire dans la région alias correspondant.



Formule pour calculer l'adresse dans la région alias :

$$\text{bit_band_address} = \text{alias_region_base} + (\text{region_base_offset} \times 32) + (\text{bit_number} \times 4)$$

2.4. Système boot

Dans les microcontrôleurs cortex M, la zone de code commence à partir de l'adresse 0x00000000 (accessible via I-bus et D-bus), tandis que la zone de la SRAM commence à partir de l'adresse 0x20000000 (accessible vis S-bus).

Selon l'état des lignes BOOT0 et BOOT1, le microcontrôleur admet trois modes de démarrage.

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

Après un RESET, le microcontrôleur commence à partir à partir de l'adresse 0x00000000, Initialise le pointeur de pile MSP, par la suite il charge l'adresse du début de l'application à partir de l'adresse 0x00000004.



2.5. Interconnexions du CPU avec les mémoires et les périphériques

Les mémoires flash et SRAM sont connectées au CPU à travers le bus AHB, tandis que les périphériques sont connectés à un ou plusieurs bus APB. Le bus AHB a également pris en charge des maîtres de bus supplémentaires, tels que des unités DMA. En ajoutant des périphériques plus complexes tels que l'Ethernet MAC et USB ayant leurs propres DMA. Avec la puissance et la complexité des périphériques, l'arbitrage des bus est devenu difficile à maîtriser.

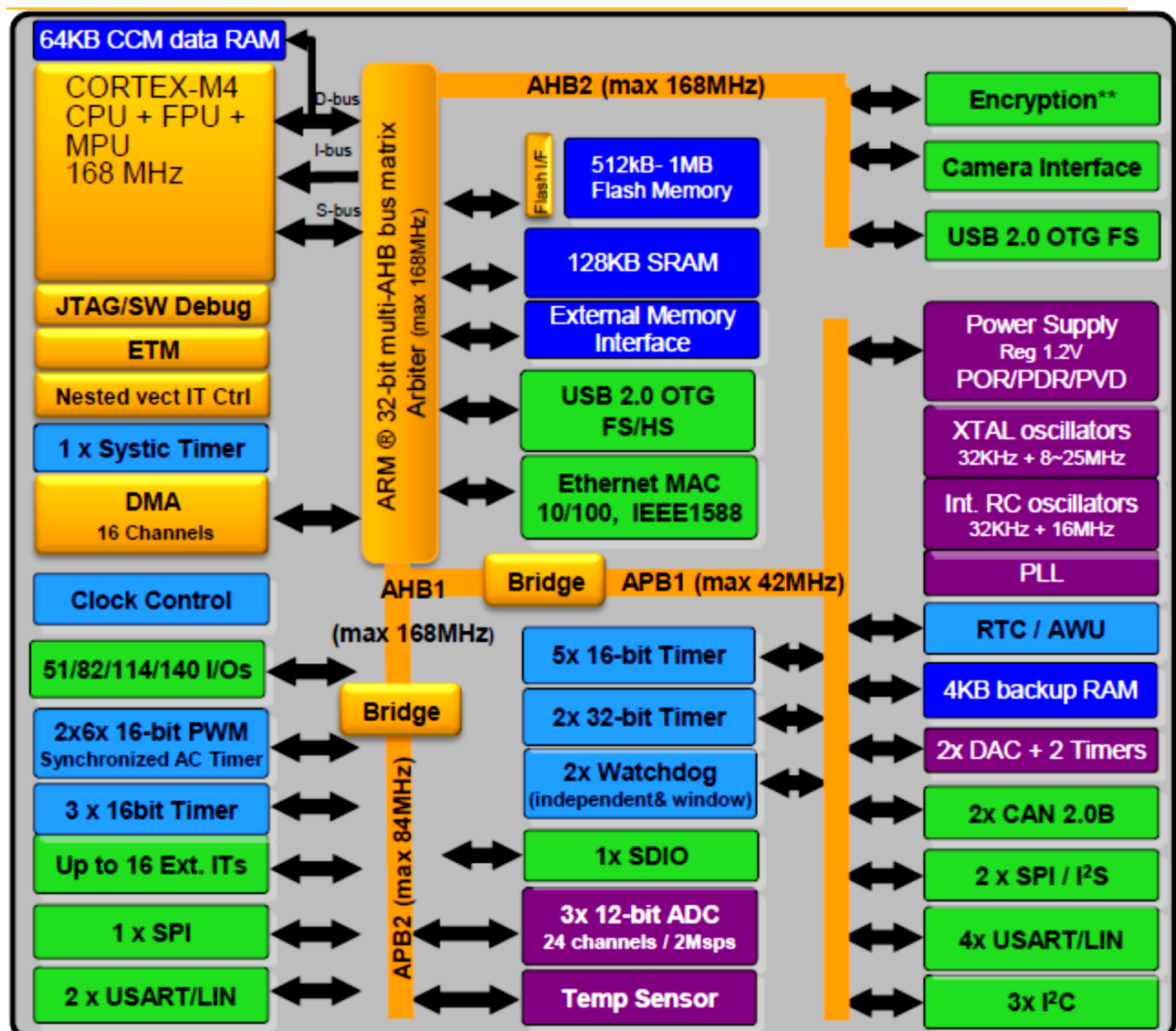
La famille de processeurs Cortex-M résout ce problème en utilisant une matrice de bus AHB. La matrice de bus AHB consiste en un certain nombre de bus AHB parallèles connectés à différentes régions du circuit. Ces zones sont définies par le fabricant lors de la conception de la puce. Chaque région est un périphérique esclave; il peut s'agir d'une mémoire flash, d'un bloc de mémoire SRAM ou d'un groupe de périphériques utilisateur sur un bus APB. Chacune de ces régions est ensuite connectée à chacun des maîtres de bus par le biais de bus AHB supplémentaires pour former la matrice de bus. Cela permet aux fabricants de concevoir des périphériques complexes avec plusieurs processeurs Cortex-M, unités

DMA et périphériques avancés, chacun avec des chemins d'accès parallèles vers les différentes ressources de périphérie.

3.STM32F4xx

3.1. Présentation

Le circuit STM32F4xx est basé sur une architecture ARM® Cortex™-M4 RISC 32 bits, fonctionnant à une fréquence maximale de 168 MHz. Il intègre une mémoire flash de 1Mo, une mémoire système SRAM de 128 Ko, une mémoire SRAM de sauvegarde de 4 Ko, une RAM couplée au core de 64Ko et une vaste gamme de périphériques d'entrées-sorties. L'intégration efficace des différents composants exige l'emploi d'un bus système à haute performance et ayant une grande bande passante pour effectuer des opérations en parallèle.



3.2. Matrice d'interconnexion

L'intégration efficace des différents composants exige l'emploi d'un bus système à haute performance et ayant une grande bande passante pour effectuer des opérations en parallèle.

Pour résoudre le problème de la bande passante, la société ARM a proposé le « Multi-layer AHB bus », sous forme d'une matrice d'interconnexion basée sur le protocole AHB. Cette matrice permet à plusieurs composants maîtres d'être interconnectés avec plusieurs esclaves.

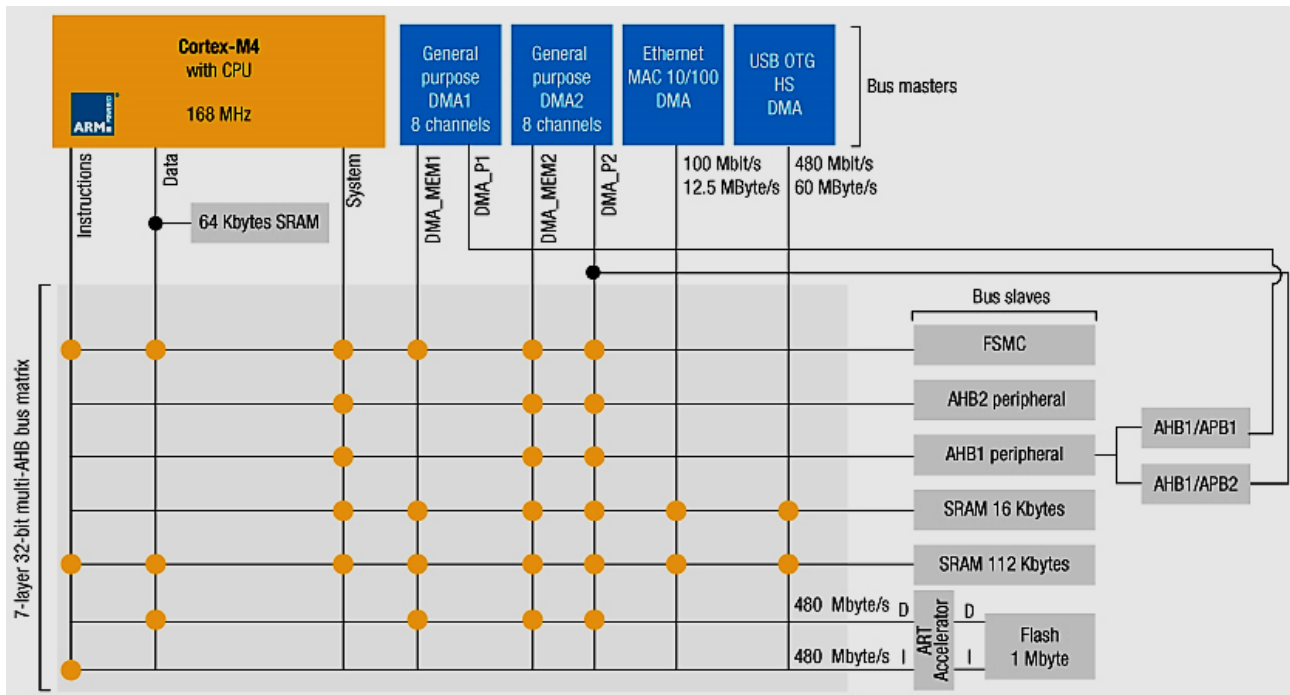
Le microcontrôleur STM32F4xx comporte huit bus maîtres et 7 esclaves :

Bus maîtres :

- Cortex®-M4 ICode, DCode et System bus
- DMA1 memory bus
- DMA2 memory bus
- DMA2 peripheral bus
- Ethernet DMA bus
- USB OTG HS DMA bus

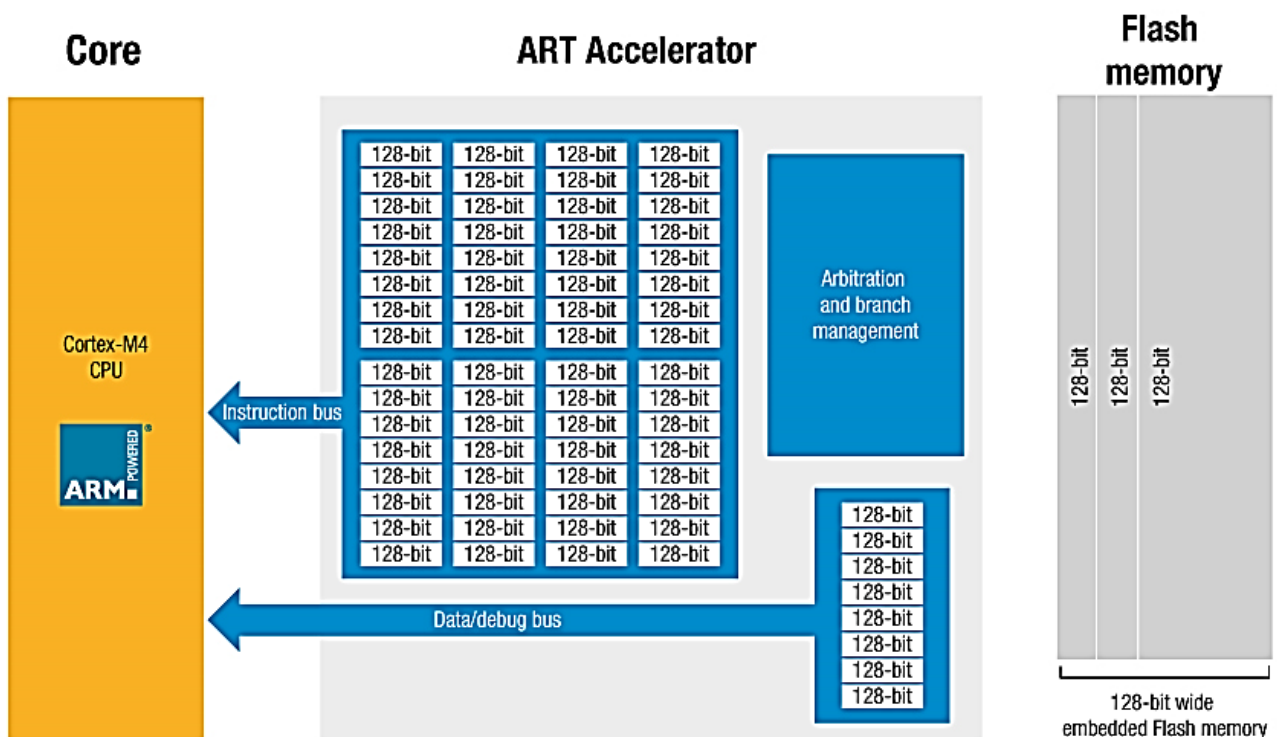
Bus esclaves :

- Internal Flash memory ICode bus
- Internal Flash memory DCode bus
- Main internal SRAM1 (112 KB)
- Auxiliary internal SRAM2 (16 KB)
- AHB1 peripherals including AHB to APB bridges and APB peripherals
- AHB2 peripherals
- FSMC



La matrice de bus permet l'interconnexion d'un maître à un esclave, ce qui permet un accès simultané et le fonctionnement efficace même lorsque plusieurs périphériques à haute vitesse fonctionnent simultanément.

3.3. Adaptive Real-Time (ART) Accelerator



3.4. Architecture du microcontrôleur STM32F4xx

L'architecture du microcontrôleur STM32F4xx est donnée par la figure suivante :

