École Nationale des Sciences de l'Informatique

Chapitre 4

Étude des microcontrôleurs STM32

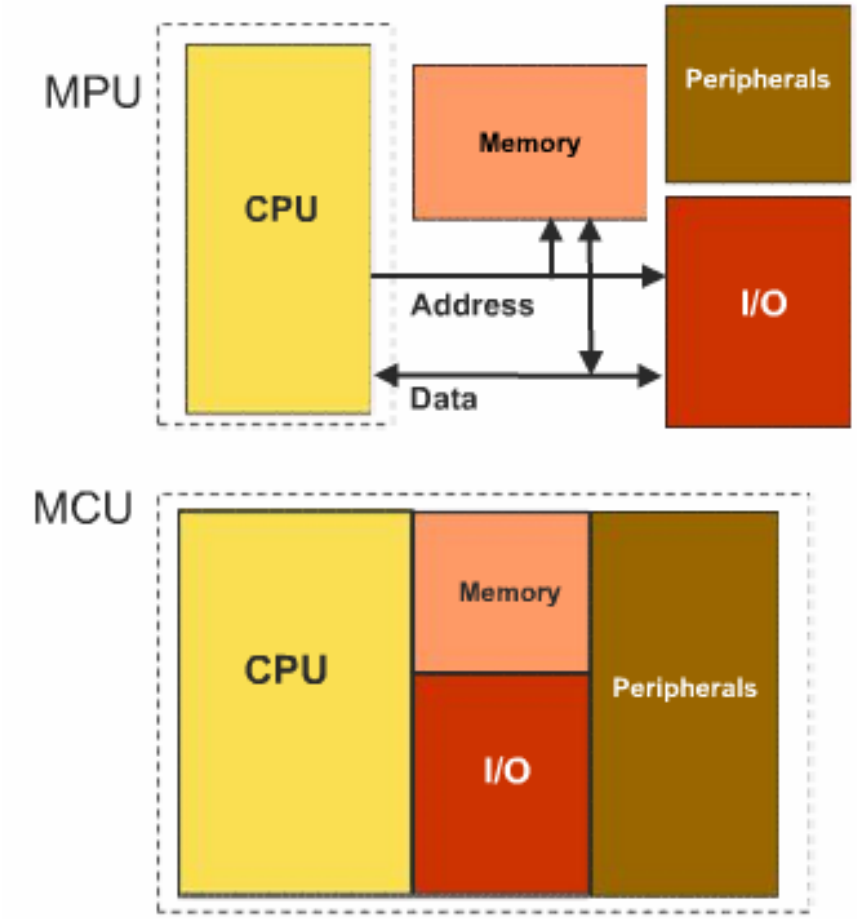
Module : Introduction aux systèmes embarqués
Niveau : II2 – Tronc commun

AU : 2011/2012

Plan

- Introduction
- Cortex-M3
- Cortex-M3 based MCUs: The STM32F10x Family
- STM32 Programming
- STM32VL Discovery Kit

Introduction

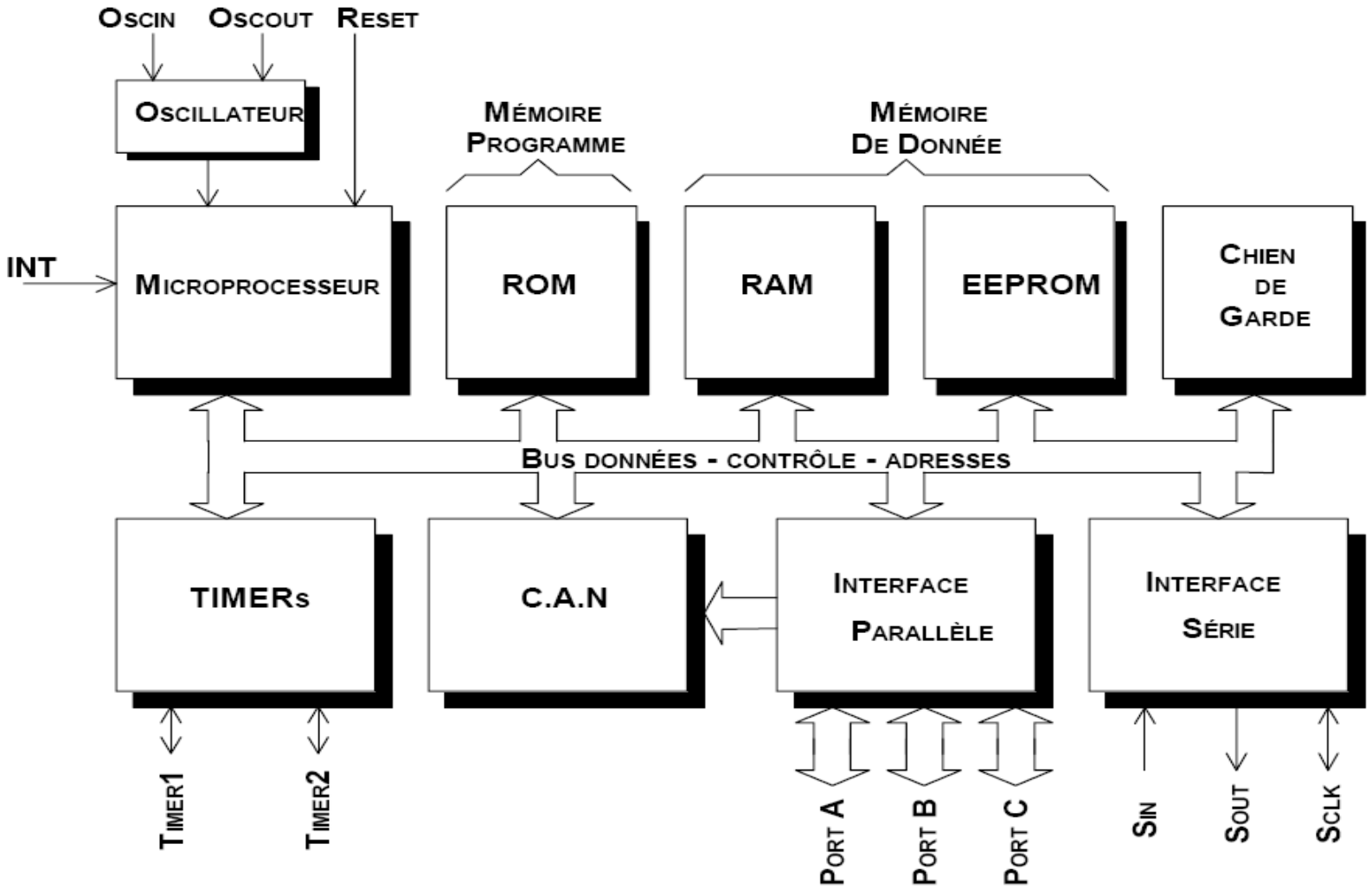


- Microprocessors (MPU): CPUs that connect to external memory and peripherals
- Microcontrollers (MCU): have CPU core with memory, I/O and peripherals integrated on-chip

Introduction

- Un microcontrôleur ou MCU (Microcontroller Circuit Unit) est un système informatique entier (CPU, mémoire, bus, contrôleur d'interruption, interfaces parallèles et série, Timers, Convertisseurs A/N et N/A) contenu intégralement dans un circuit intégré unique.

Introduction:Architecture de base



Domaines d’applications

- Ces circuits à prix réduits sont d’une grande utilité pour les applications embarquées (Machines à laver, systèmes GPS, climatiseurs, cartes à puce, automates programmables, contrôleurs d’ascenseurs, variateurs de vitesse, etc..).

Introduction:Evolution

- Les microcontrôleurs 32 bits ont énormément progressé lors de la dernière décennie et sont passées de moins de 1% en 2000, à 30% en 2006 pour atteindre les 38 % en 2009

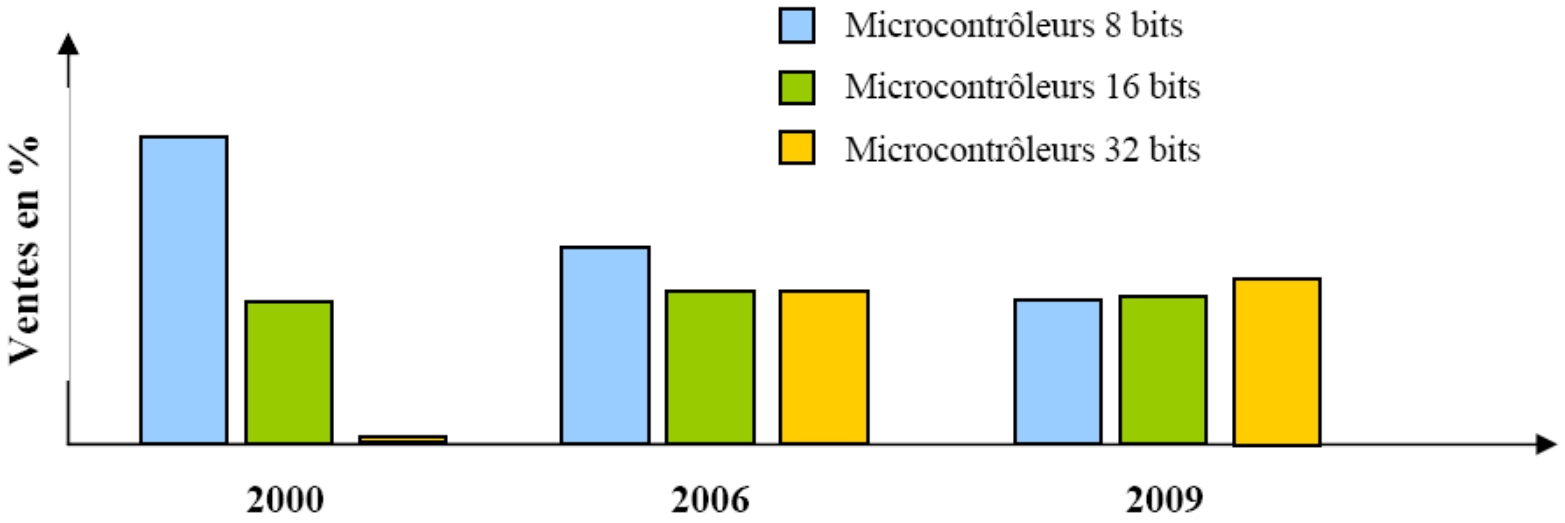


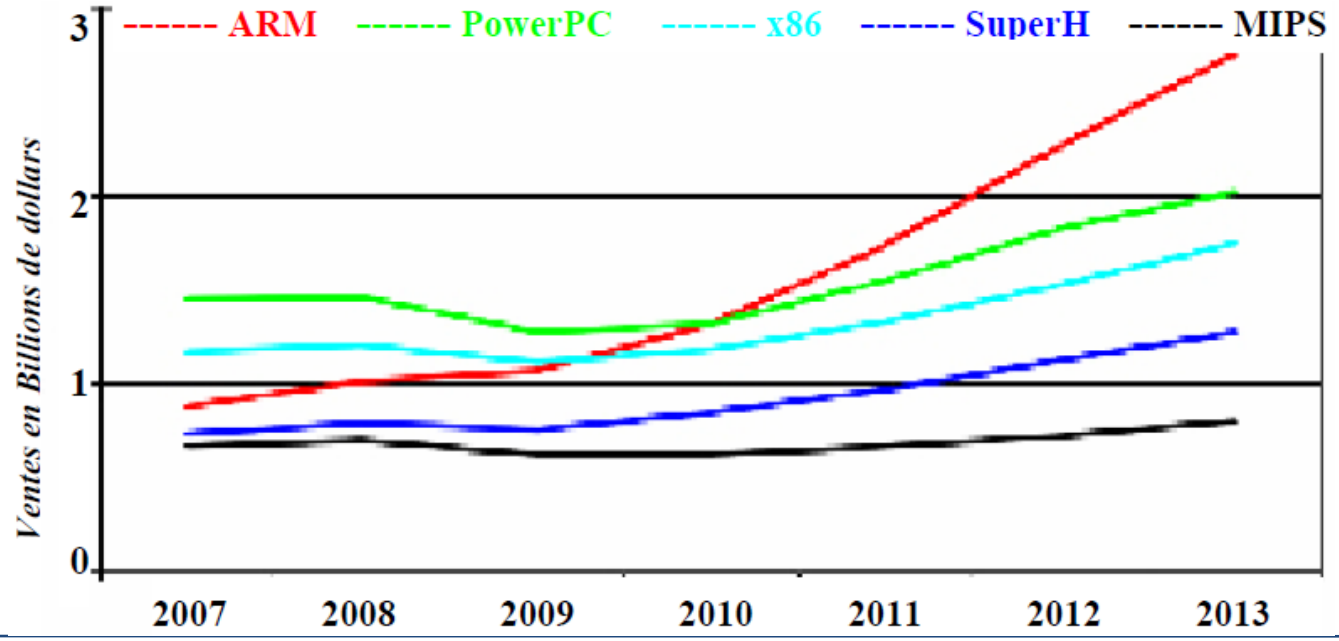
Figure 1 : Evolution des ventes en % des microcontrôleurs 8, 16 et 32 bits

Introduction

- La majorité des fabriquants de microcontrôleurs optent pour l'intégration d'un CPU préconçu, sous forme d'un IP « Intellectual Property », dans leurs circuits et lui ajoutent les mémoires et les entrées/sorties nécessaires pour les applications ciblées.
- Parmi les CPUs (IP) les plus utilisés dans les microcontrôleurs 32 bits, on trouve les cœurs de la société ARM (Advanced Risc Machines), qui grâce à son nouveau cœur Cortex s'est imposé comme leader des fournisseurs de cœurs 32 bits pour les applications embarquées.

Pourquoi ARM?

- Les processeurs ARM sont surtout utilisés dans les systèmes portables à cause de:
  - Sa faible consommation d'énergie
  - Ses performances raisonnables



Introduction

- L'architecture et le jeu d'instructions du coeurs ARM ont évolué depuis la première version ARM-v1 jusqu'à la version actuelle ARM-v7, appelée également Cortex.

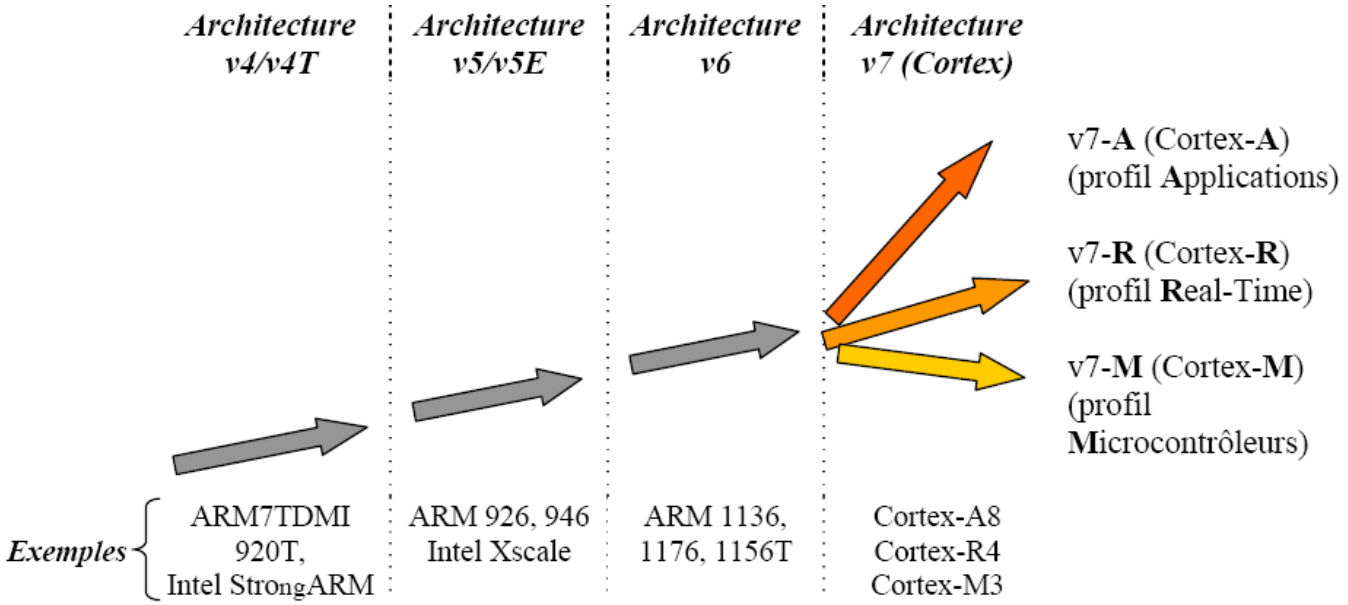
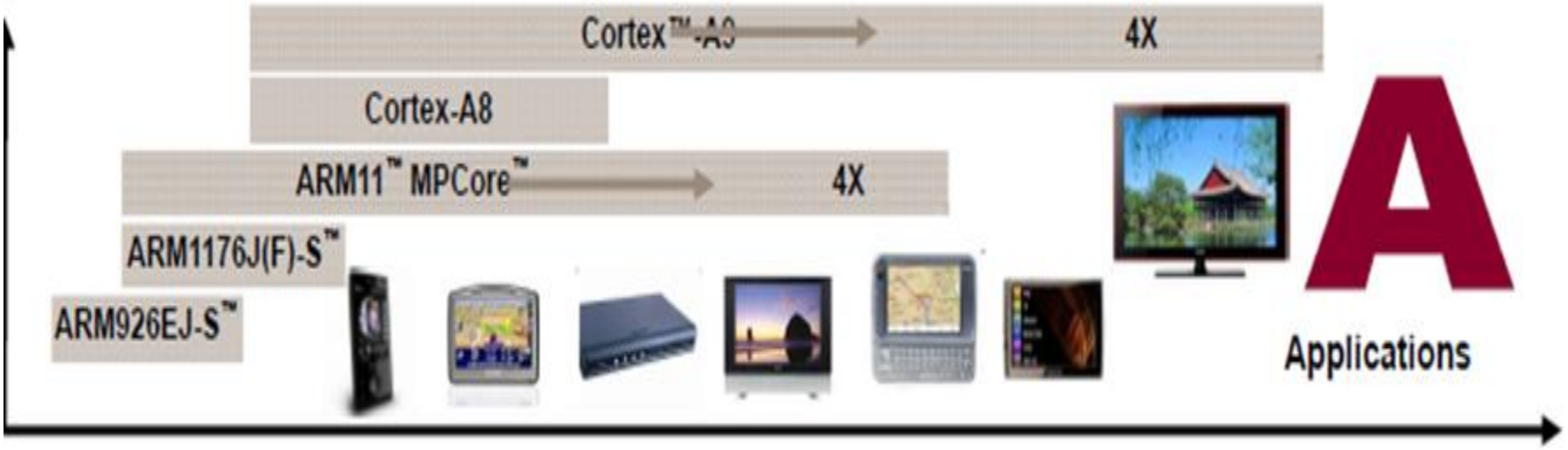


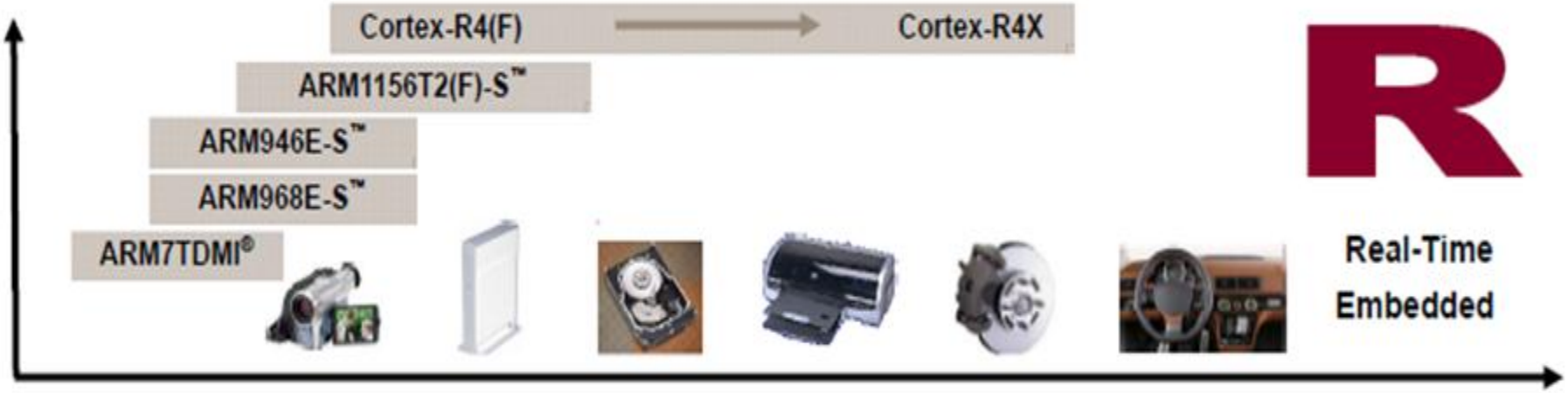
Figure 3 : Evolution des architectures des cœurs ARM

Introduction



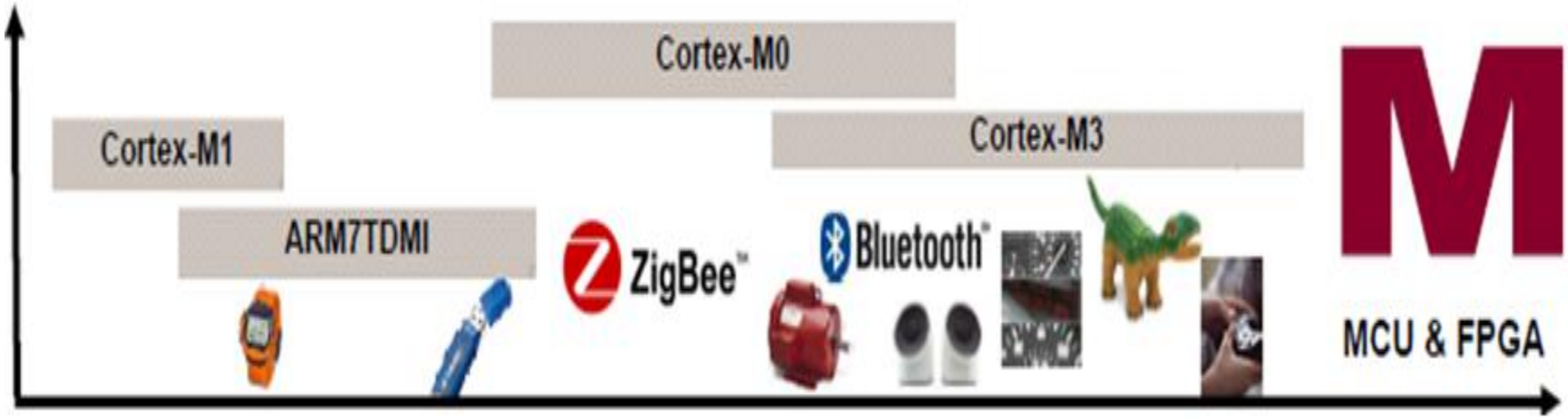
Le profil A (séries Cortex-A) : destiné pour faire tourner des applications complexes telles que les systèmes d'exploitation embarqués (linux, Windows) nécessitant une puissance de traitement élevée et un système de gestion de mémoire virtuelle (MMU: Memory Mangement Unit).

Introduction



Le profil R (séries Cortex-R) : destiné principalement aux applications temps réel à contraintes très sévères et exigeant une haute fiabilité et un temps de réponse faible

Introduction



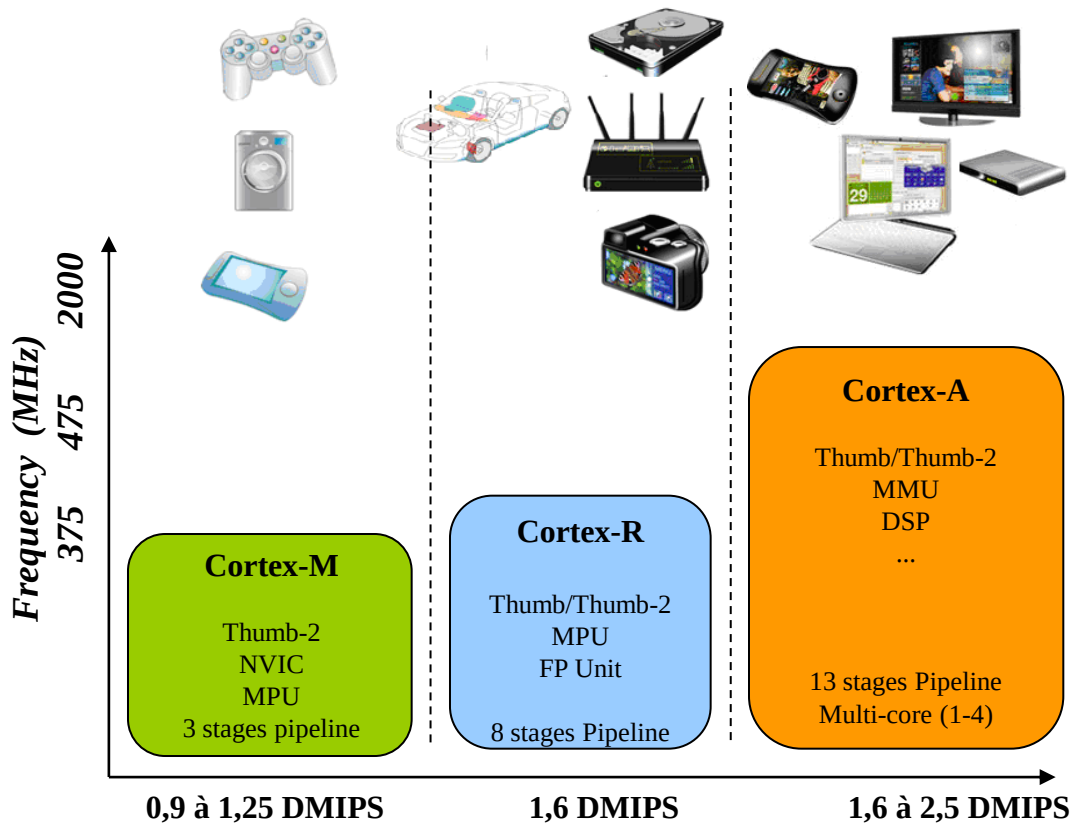
Le profil M (séries Cortex-M) : optimisé pour des applications nécessitant une basse consommation en énergie, un coût réduit et un comportement déterministe pour le traitement des interruptions. C'est le profil qui est destiné à être intégré dans les microcontrôleurs (utilisés surtout pour la commande des machines).

Cortex profiles & applications

Cortex-X N

X :Profile (A,R,M)

N: Performance level (0..9)



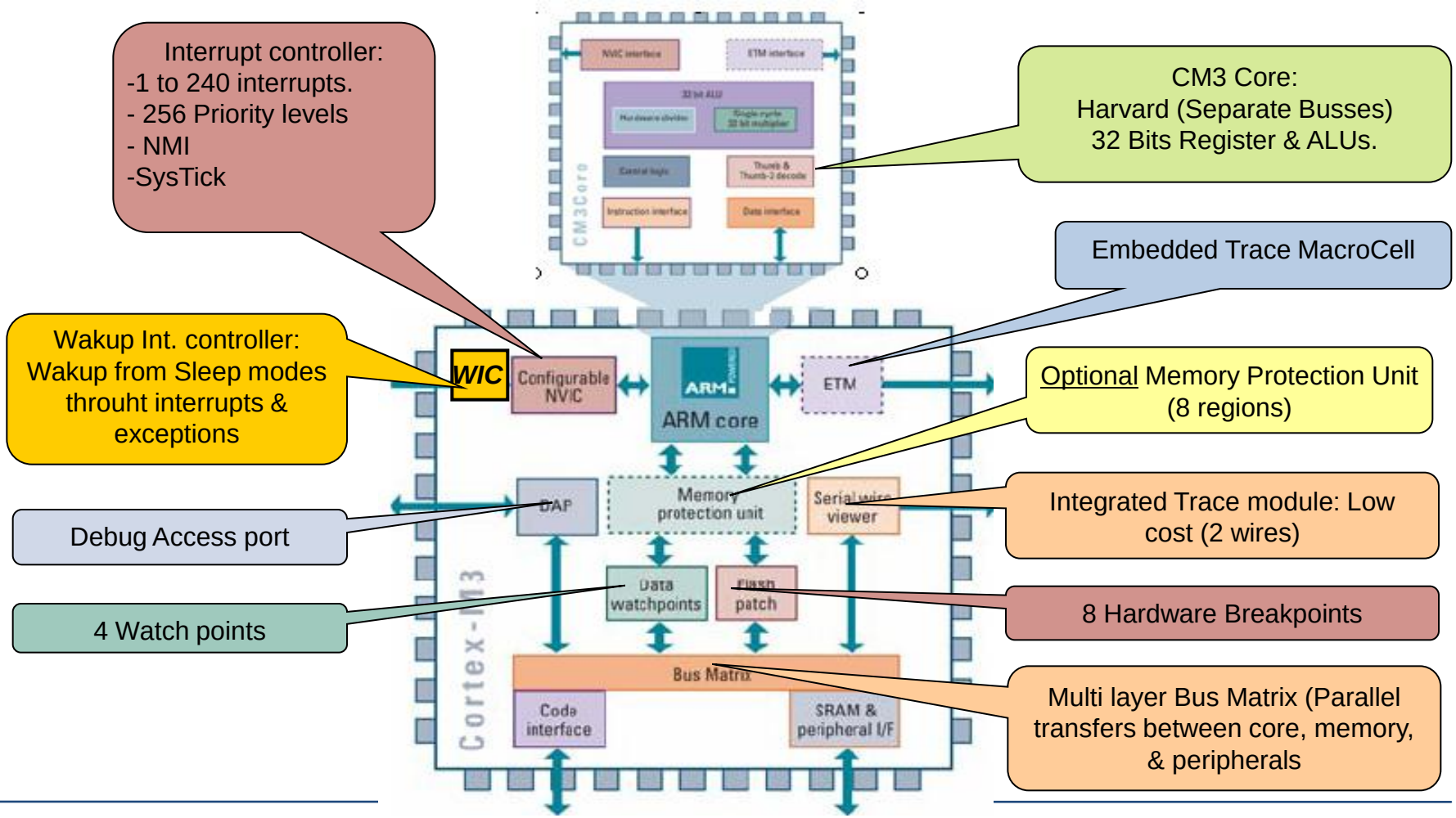
Manufacturers	Cortex-M3
	STM32 L1xx STM32 F1xx STM32 F2xx
Texas Instruments	Stellaris3x
NXP	LPC17x, LPC3x
Atmel	SAM3x
freescale	LM3S8x
Analog Devices	ADuCRF101
Toshiba	TX03
Samsung	S3FM

Outline

- Introduction
- Cortex-M3
- Cortex-M3 based MCUs: The STM32F10x Family
- STM32 Programming
- STM32VL Discovery Kit

Cortex-M3 Processor

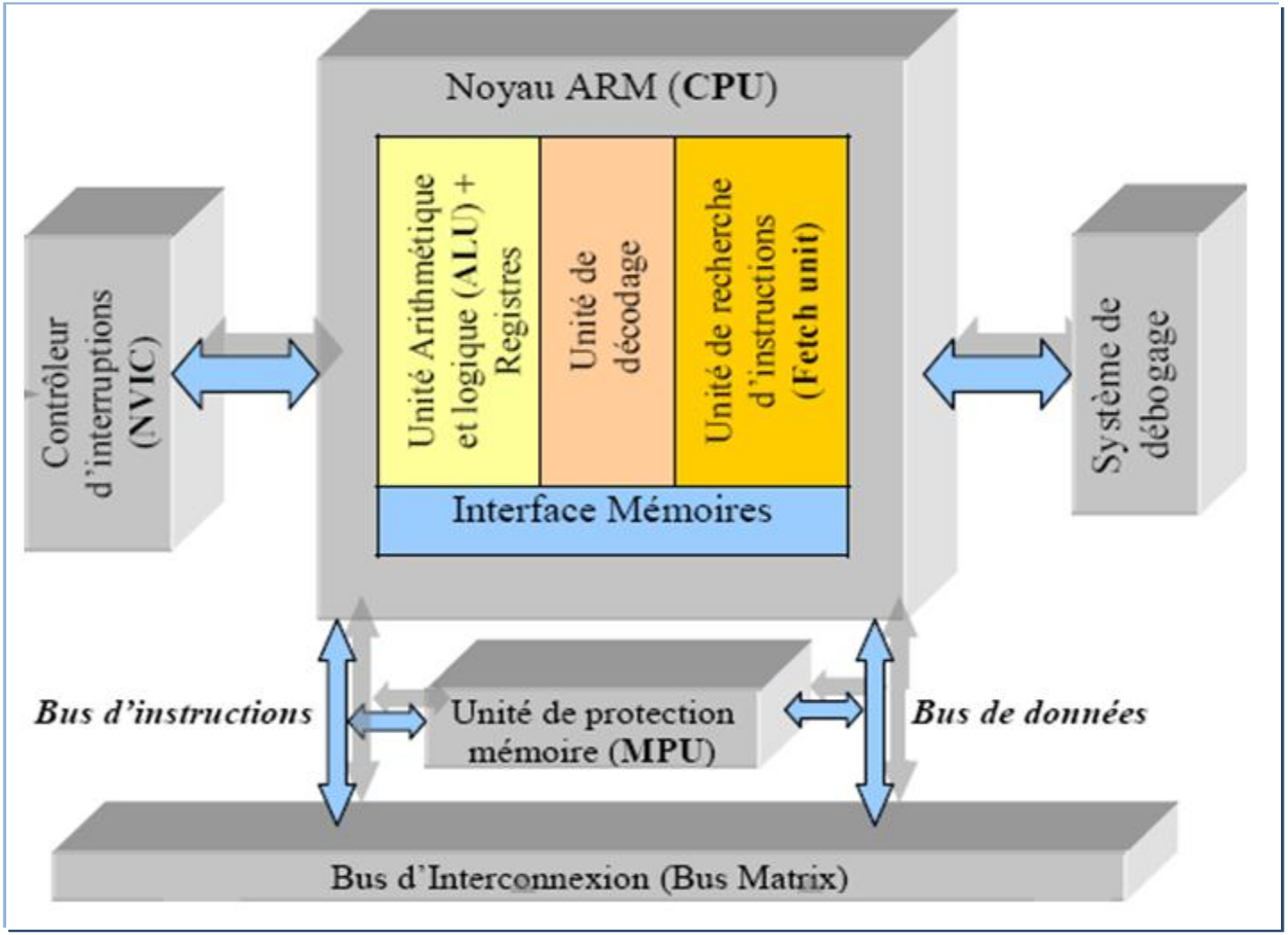
- Hierarchical processor integrating core and advanced system peripherals



Cortex-M3 Processor

- C’est un processeur 32 bits.
- Il possède une architecture Harvard, c'est-à-dire qu’il contient deux bus distincts pour le transfert des données et des instructions. Ceci permet d’accéder en même temps aux instructions et aux données.
- Il se base sur un pipeline à 3 étages (Fetch, Decode et Execute)
- Il possède aussi une UAL 32 bits avec:
  - des divisions matérielles (SDIV et UDIV) qui nécessitent de 2 à 3 cycles horloge.
  - des multiplications 32 bits qui se déroulent en un cycle horloge.
- L’espace d’adressage du Cortex-M3 peut atteindre la taille de 4 GO.

Cortex-M3 Processor



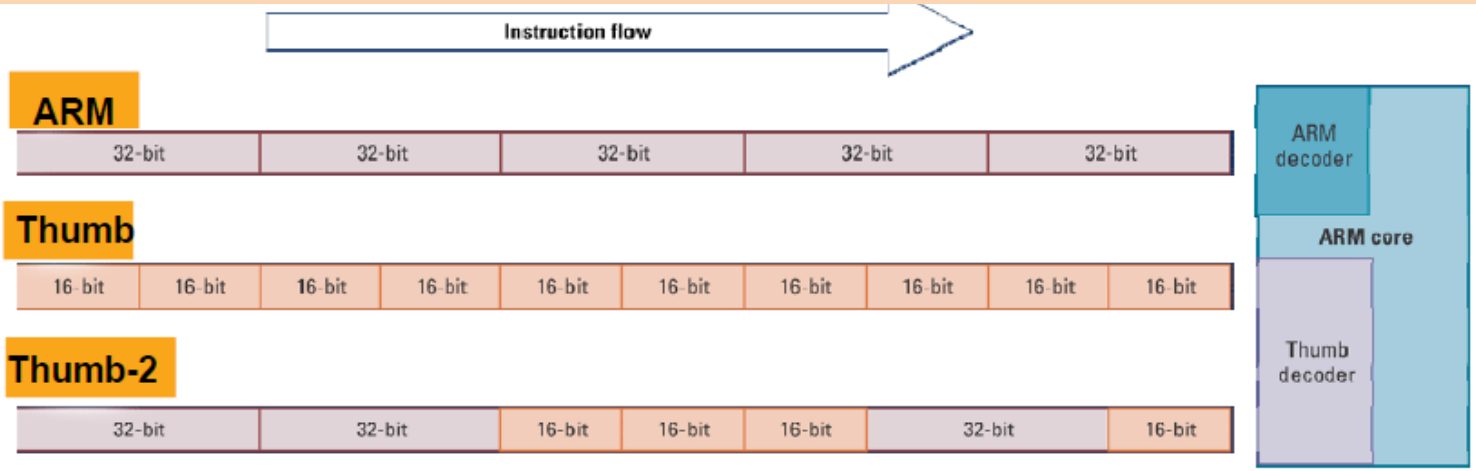
Cortex-M3 Processor

- Il intègre également un certain nombre de composants de débogage qui permettent de fixer des points d'arrêt (breakpoints) et d'observation (watchpoints) ainsi que le suivi des instructions (instruction trace). Ce qui permet une bonne visibilité du processeur et de la mémoire à travers le port JTAG ou le port SWD(a 2-pin *Serial Wire Debug*).

Cortex-M3: Caractéristiques

Thumb®-2 and traditional Thumb

L'une des particularités du Cortex-M3 est son jeu d'instruction ***thumb-2.***
En effet, étant donnée que les instructions ARM de 32 bits occupent un espace mémoire important, et que les mémoires utilisées étaient généralement de petite taille, ARM a introduit depuis son architectures ARMv4 un ensemble d'instructions sur 16 bits constituant un jeu d'instruction nommé Thumb. Ainsi, cette solution a apporté un gain en occupation mémoire. Mais, elle a introduit une baisse de performances



Cortex-M3: Caractéristiques

Pour le Cortex-M3, le jeu d'instruction Thumb 16 bits a été gardé, mais le jeu d'instructions ARM 32 bits a été complètement abandonné au profit d'un nouveau jeu d'instructions Thumb 32 bits. L'ensemble ainsi obtenu des instructions Thumb 16 et 32 bits est appelé **Thumb-2.** D'après ARM, il permet d'optimiser l'utilisation de l'espace mémoire sans dégrader les performances.

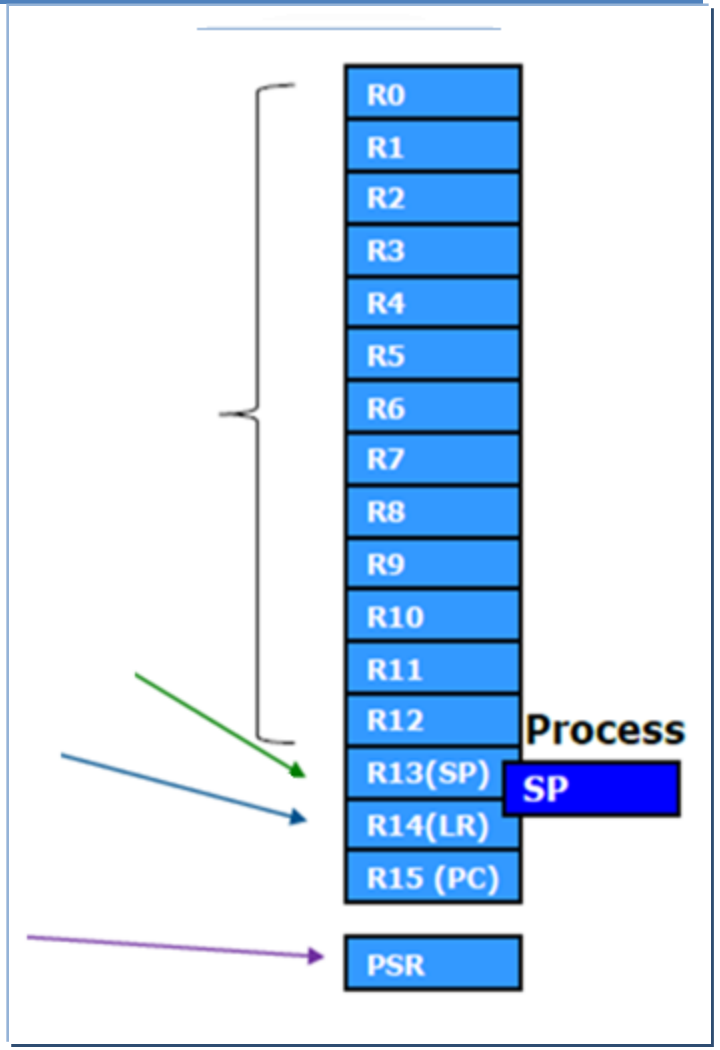
Cortex-M3 : Register Set

12 registres à usage général qui peuvent être utilisés pour l'exécution de n'importe quelle opération (arithmétiques, logiques, transfert).

R13 pointeur de pile:

R14 : il reçoit l'adresse de retour lors d'un appel à un sous programme

PSRs : registre d'état de programme



Cortex-M3 : xPSR Register

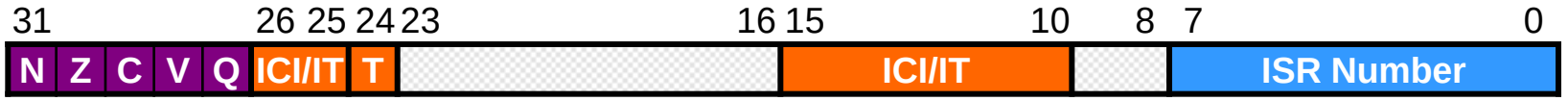
- ***xPSR (Program Status Registers) : Registres d'état du programme qui sont au nombre de 3***
- *- ASPR (Application PSR): Registre d'état relatif à l'exécution du programme.*
- *- ISPR (Interrupt PSR) : Registre d'état relatif à l'interruption en cours.*
- *- ESPR (Execution PSR) : Registre d'état qui donne des informations sur l'instruction en cours*
- Ces 3 registres 32 bits peuvent être accédés séparément ou bien être superposés pour construire un registre unique 32 bits : xPSR.
- xPSR: sont stokés dans la pile en cas d'excéption ou d'interruption.

Cortex-M3 Core: xPSR Register

3 Registres d'états

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
Application PSR APSR	N	Z	C	V	Q											
Interrupt PSR IPSR												Exception Number				
Execution PSR EPSR						ICI/IT	T			ICI/IT						

Les trois registres combinés dans un seule registre read/write



Cortex-M3 Core: xPSR Register

- Condition code flags
  - N = bit de signe
  - Z = bit de Zero
  - C = bit de retenu
  - V = débordement (overflow)
  - Q =ce bit est utilisé par les instructions SSAT et USAT pour indiquer que la variable a atteint sa valeur min ou max.
- xPSR: est stoké dans la pile en cas d'excéption ou d'interruption.

Cortex-M3 Core: xPSR Register

- IT/ICI Bits
  - IT: IF-THEN condition code: Si la condition est vrai le bit IT sera positionné à 1 pour permettre au processeur d'éxécuter les 4 instructions suivante. Si non ses quatre instructions vont passer à travers le pipeline comme étant des NOP.
  - ICI :Interrupt Continue information
- ISR Number
  - ISR contient le numéro de l'interruption qui a était interrompu.

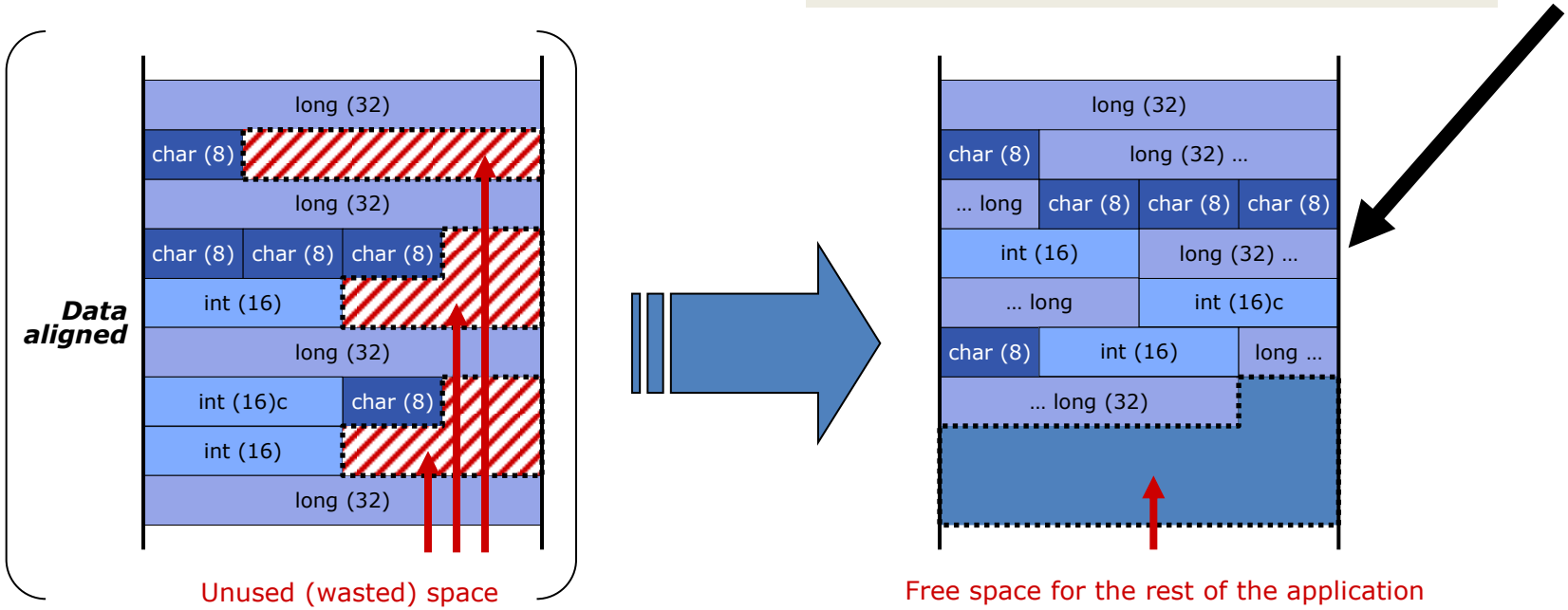
Cortex-M3: Gestion de la Mémoire de Donnée

- Cortex-M3 inclue deux technologies qui permettent d'améliorer considérablement l'utilisation de la mémoire de données (SRAM):
  - Non alignement
  - Bit banding

Cortex-M3: Non alignement des donnés

Il fait l'alignement des données: **l'adresse de la donnée doit etre multiple de sa taille**

Il supporte le mode non alignement : **les données peuvent etres stokés n'importe ou dans la mémoire**



Reduces SRAM Memory Requirements

Cortex-M3: Non alignement des donnés

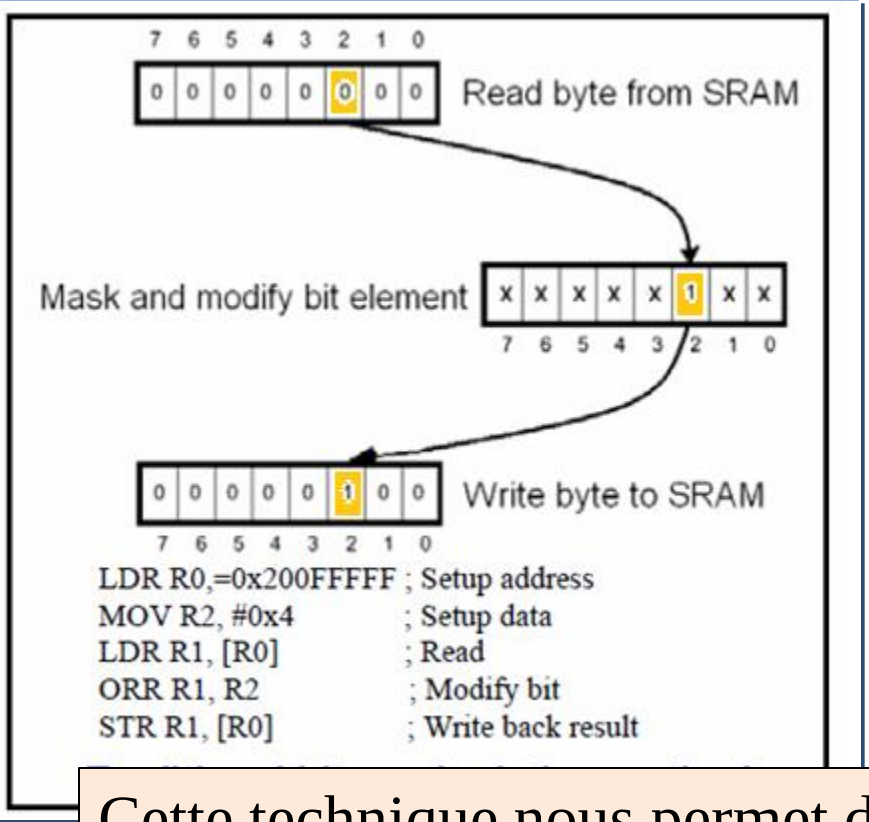
- Le mode non alignement n’est pas utilisé dans le cas :
  - Des opération sur la pile (PUSH/POP).
  - Bit-band operations
- Dans le cas d’un transfert en mode non alignement , le transfert d’une information peut prendre plusieurs cycles d’horloge ce qui entraine une dégradation des pérformences.

Unaligned access:

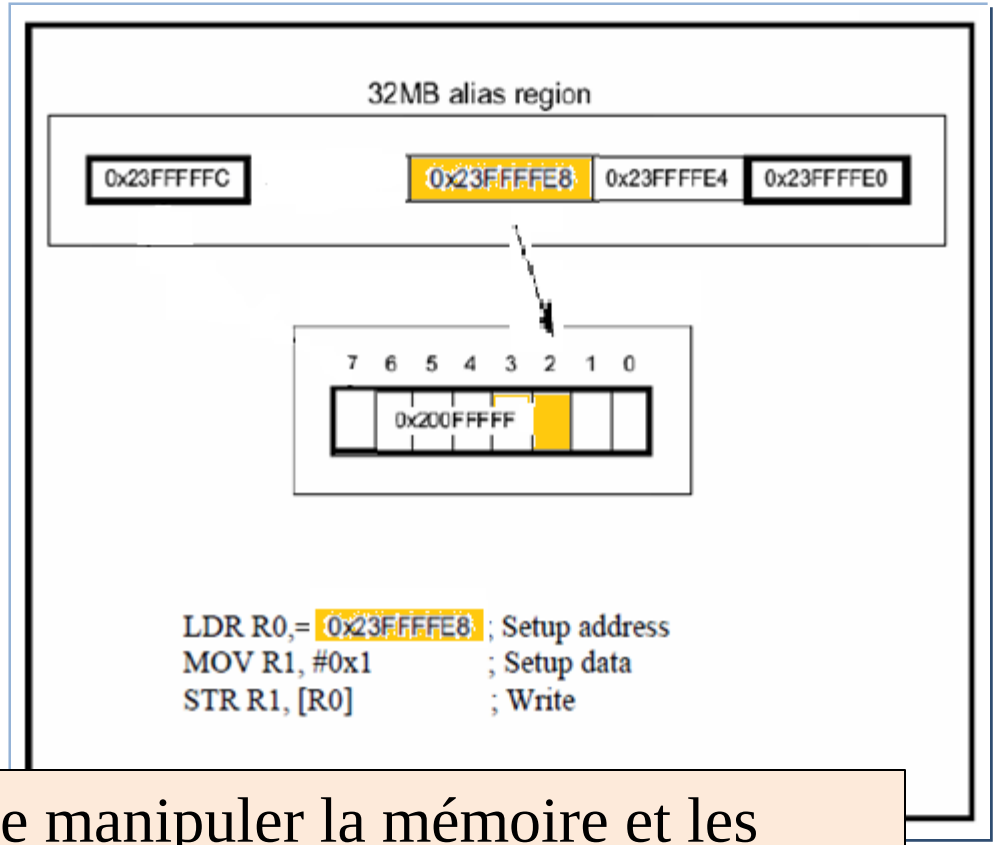
- Une meilleure utilisation de la mémoire, mais on aura dégradation de la performance

Cortex-M3: Bit Banding

Methode de manipulation de bit traditionelle



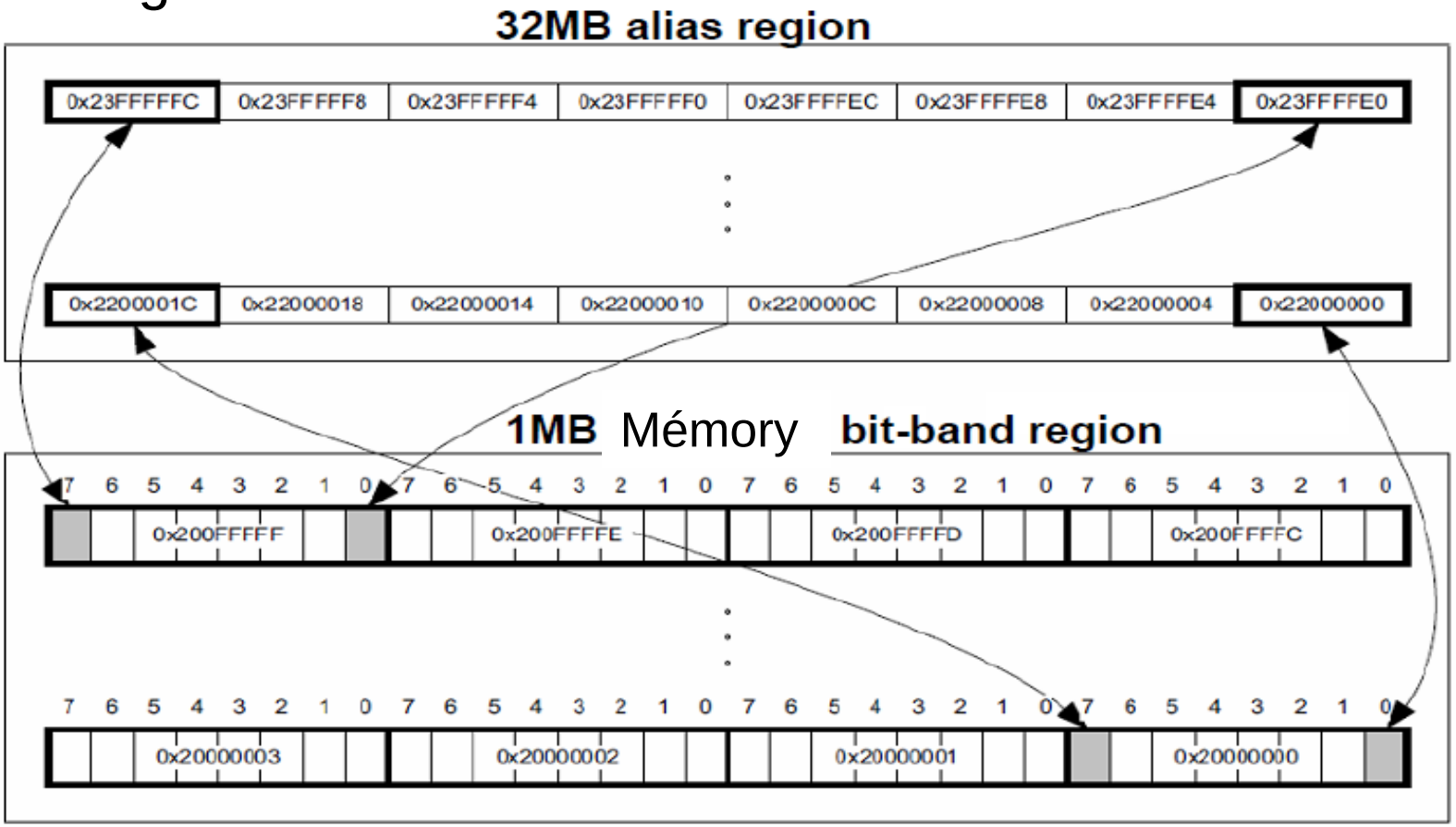
Methode du bit Banding



Cette technique nous permet de manipuler la mémoire et les périphériques au niveau du bit.

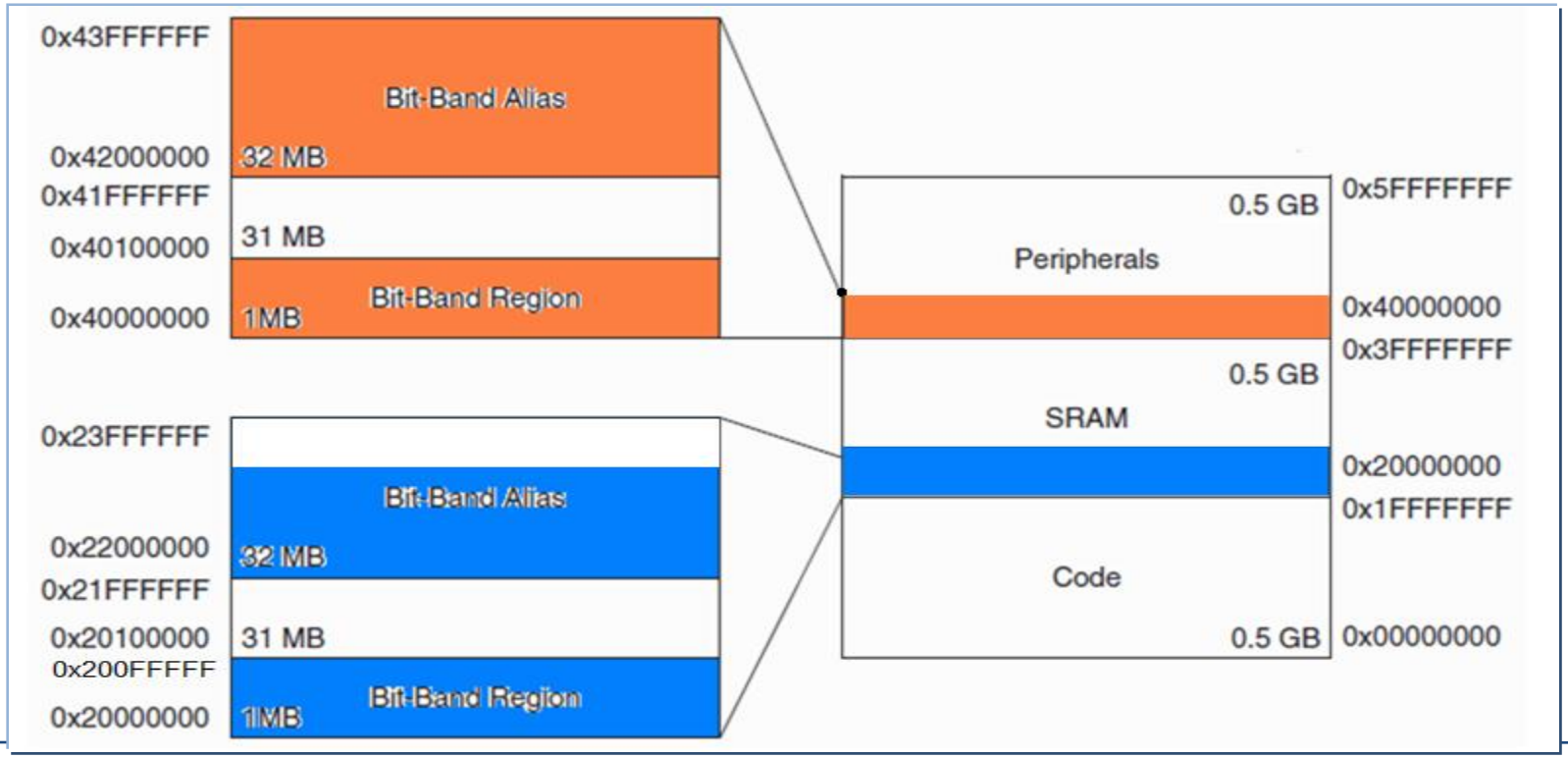
Cortex-M3: Bit Banding

En pratique pour modifier un bit dans la zone bit band mémoire ou périphérique, on doit calculer l'adresse du mot correspondant dans la région bit band alias.

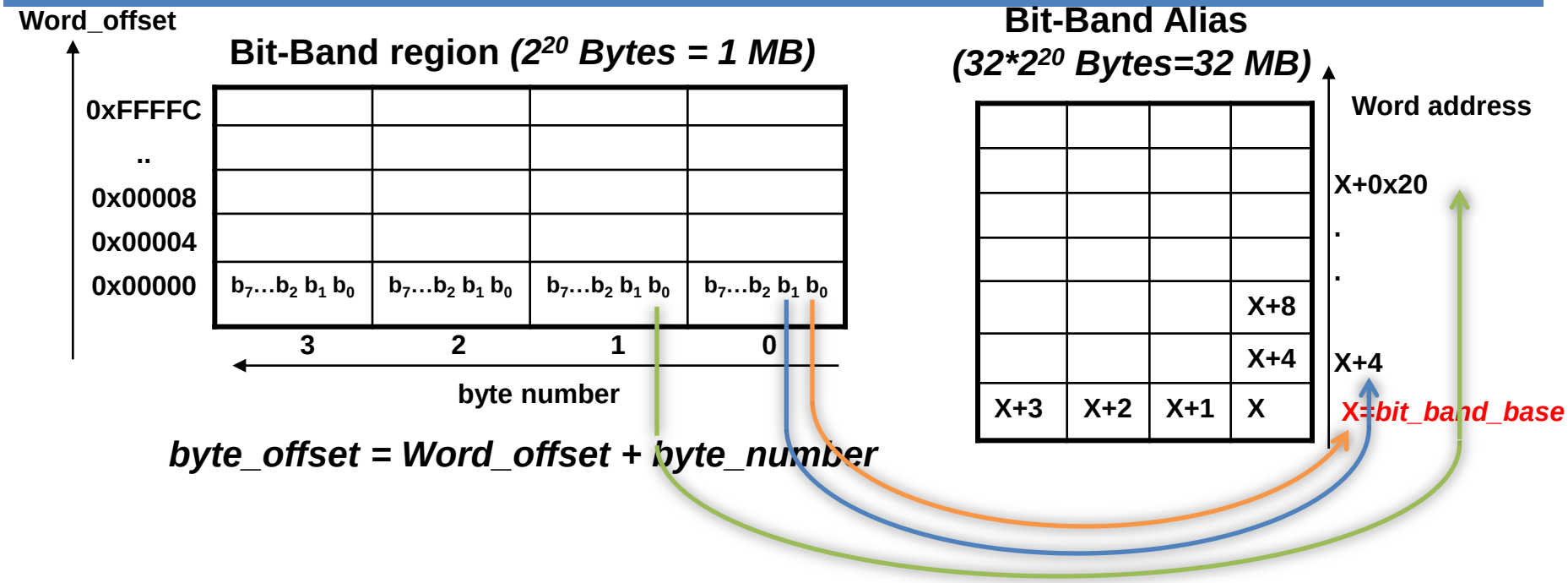


Cortex-M3: Bit Banding

A Chaque bit de la région bit band (1MByte) on fait correspondre l'adresse d'un mot de 32bit dans une zone de 32 MByte appelé Bit Band Alias (zone Virtuelle). Ainsi en forçant à 1 ou à 0 l'adresse du mot dans la zone bit band alias on force le bit correspondant dans la région bit band



Cortex-M3: Bit Banding



Pour accéder au premier bit de la région de la bande, nous devons passer par le X adresse de mot de la région d'alias.

Pour accéder au deuxième bit de la région de la bande, nous devons passer par l'adresse du mot X+4 de la région d'alias. Et ainsi de suite .....

Cortex-M3: Bit Banding

- **La formule qui permet de trouver l'adresse du bit dans la région d'Alias est la suivante:**

$$bit\_word\_addr = bit\_band\_base + (byte\_offset \times 32) + (bit\_number \times 4)$$

bit_word_addr: C'est l'adresse du mot dans la région de la mémoire Alias qui correspond au bit ciblé

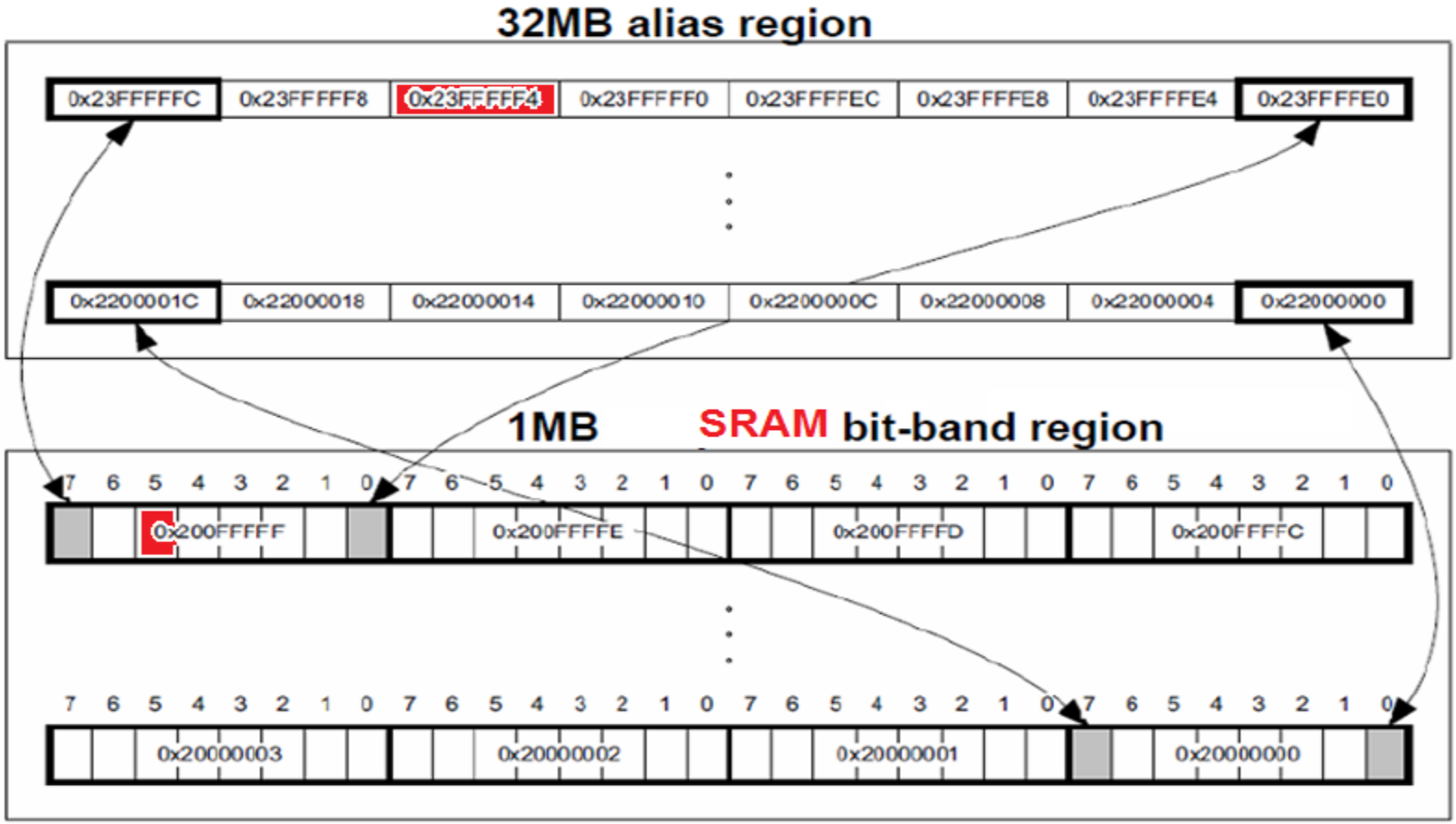
bit_band_base : C' est l'adresse de départ de la région d'alias (0x22000000 ou 0x42000000)

byte_offset : c'est le numéro de l'octet dans la région de bit de bande qui contient le bit ciblé.

bit_number : est la position du bit cible (0-7).

Cortex-M3: Bit Banding
Exercice 1

Comment faire pour trouver l'adresse du mot correspondant au bit 5 de l'octet situé à l'adresse 0x200FFFFF dans la région alias.



Cortex-M3: Bit Banding
Exercice 1

Example -1:

Comment faire pour trouver l'adresse du mot correspondant au bit 5 de l'octet situé à l'adresse 0x200FFFFF dans la région alias.

Solution:

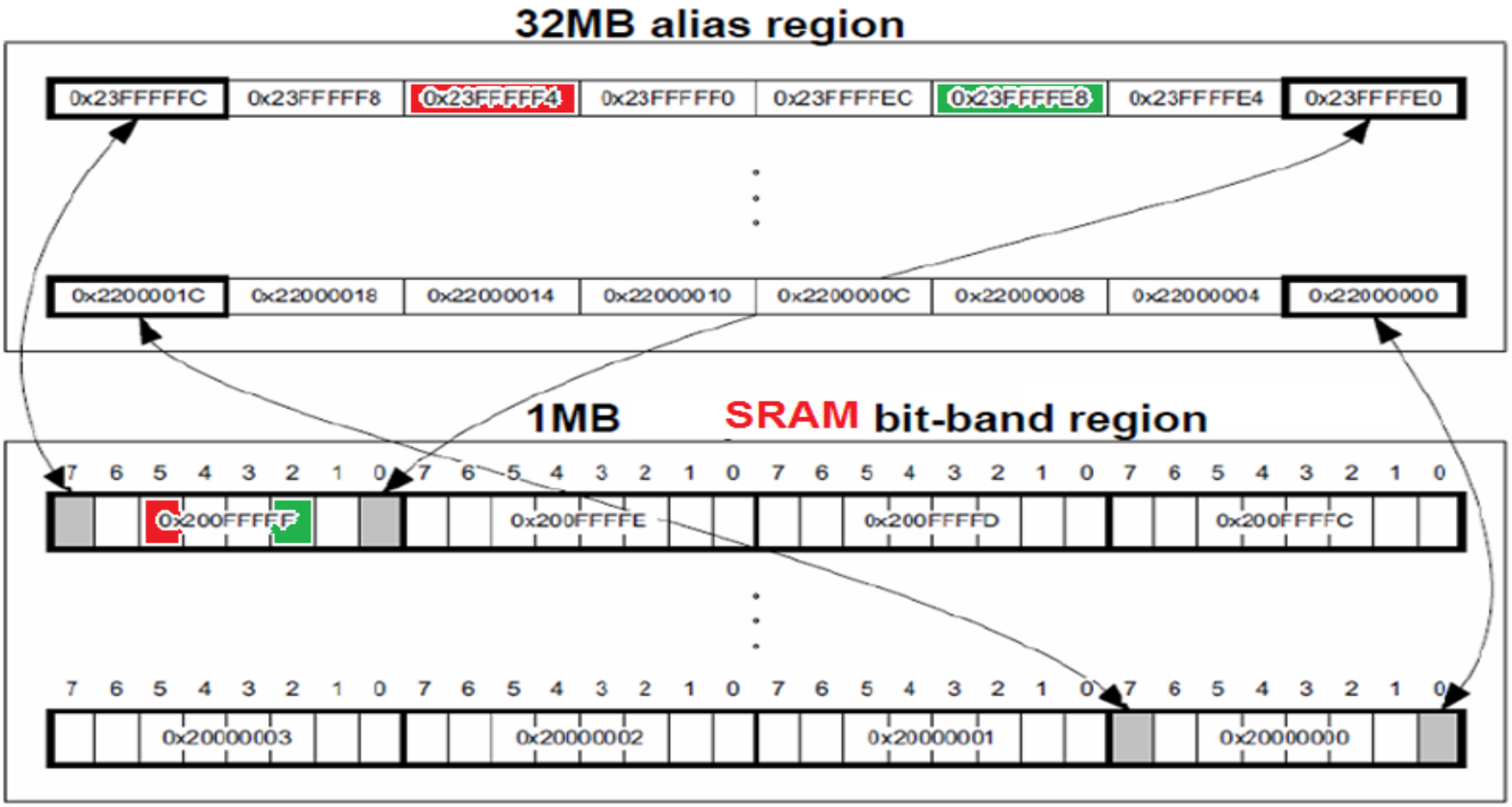
Nous allons accéder à la region bit banding de la Memoire qui commence à l'adresse 0x 0x20000000 et qui est mappé dans la région bit band Alias dont l'adresse de base bit_band_base = 0x22000000

Byte_offset = 0x200FFFFF - 0x20000000 = 0x000FFFFF

***bit_word_addr* = 0x22000000 + (0x000FFFFF *20) + (5 * 4) = 0x23FFFFF4**

Cortex-M3: Bit Banding
Exercice 2

A quel bit on va accéder lors de la lecture du mot d'adresse 0x22006008



Cortex-M3: Bit Banding
Exercice 2

Example -2:

A quel bit on va accéder lors de la lecture du mot d'adresse 0x22006008

Solution:

Nous allons accéder à la zone bit band Alia de la mémoire dont l'adresse de base est bit_band_base =0x22000000 et qui correspond à la région bit banding qui commence à l'adresse 0x 0x20000000

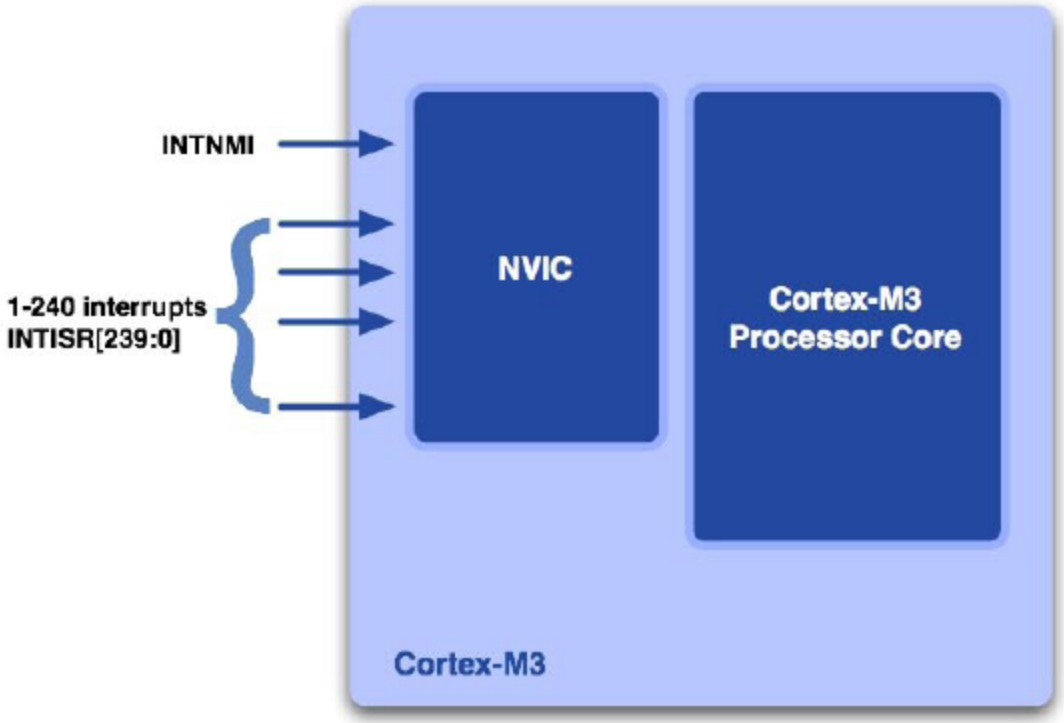
→ **0x22000000 + (byte_offset*32) + (bit_number * 4) = 0x22006008**

(byte_offset*32) + (bit_number * 4) = 0x6008

→ **byte_offset = 0x300 et bit_number =2**

Cortex-M3: Déroulement des Interruptions

Le processeur Cortex-M3 intégre un controlleur d'Interruption Evolué NVIC (Nested Vector Interrupt Controller)



Cortex-M3: Déroulement des Interruptions

- Le NVIC est capable de gérer jusqu'à 240 interruptions générées par des périphériques externes au Cotex-M3 et dont le niveau de priorité peut être dynamiquement fixé parmi 256 niveaux possibles. En plus de ces interruptions, le NVIC gère un certain nombre d'interruption déclenchées par des périphériques internes du Cortx-M3 (Bus AHB, MPU, Timer SysTick, etc..) ou par des erreurs au niveau du programme même.
- Un temps de réponse déterministe ainsi que la possibilité de préemption des interruptions grâce à l'utilisation de plusieurs niveaux de priorité. Ceci permet de supporter les applications temps réel.

Cortex-M3: Déroulement des Interruptions

- Exceptions Asynchrones = Interruptions: dues à des événements externes qui n'ont pas de relation avec l'exécution des instructions du processeur, ils sont en général associés aux signaux matériels (exemple.
  - Signal toggle (P I/O ports).
  - Data receive (Serial peripherals)
  - A/D conversion finished (DAC)
  - ...
- Exceptions Synchrones = Exceptions :provoquées par des évènements internes comme l'exécution des instructions du processeur exemple:
  - Erreur d'accès mémoire
  - Division par zero
  - Depassement de capacité
  - ...

Cortex-M3: Déroulement des Interruptions

Nr	Type d'exception	Niveau de priorité par défaut	Description
1	Reset	-3 (maximum, figé)	Déclenchée quand un signal de reset est détecté.
2	NMI	-2 (figé)	Il s'agit de l'entrée Non Masquable Interrupt qui est généralement connectée au WatchDog.
3	Hard Fault	-1 (figé)	Erreur de matériel pour lequel aucune routine n'a été prévue
4	MemManage Fault	0 (paramétrable)	Déclenchée en cas d'une tentative d'accès à une zone mémoire non autorisée.
5	Bus Fault	1 (paramétrable)	Générée quand l'interface du bus AHB reçoit une erreur.
6	Usage Fault	2 (paramétrable)	Déclenchée par une erreur au niveau d'un programme (exple : division par 0).
7-10	Réservé		
11	SVCall	3 (paramétrable)	Dans le cas d'un appel à un service système.
12	Debug Monitor	4 (paramétrable)	Déclenchée quand un point de break (ou watch) est atteint ou bien quand le débogage est activé.
13	Réservé		
14	PendSV	5 (paramétrable)	
15	SYSTICK	6 (paramétrable)	Relative au Timer interne SYSTICK
16... *....* *256*	Interrupt # 0 ..... Interrupt # 240	7 (paramétrable) ..... 247 (paramétrable)	Il s'agit d'événements déclenchés par les autres périphériques externes ajoutés par le fabricant du microcontrôleur.

Cortex-M3: Déroulement des Interruptions

L'ensemble des adresses des routines relatives aux différentes sources d'interruptions forme la table de vecteurs d'interruptions**.** Dans le Cortex-M3, la table de vecteurs commence à partir de l'adresse 0 de la zone code :

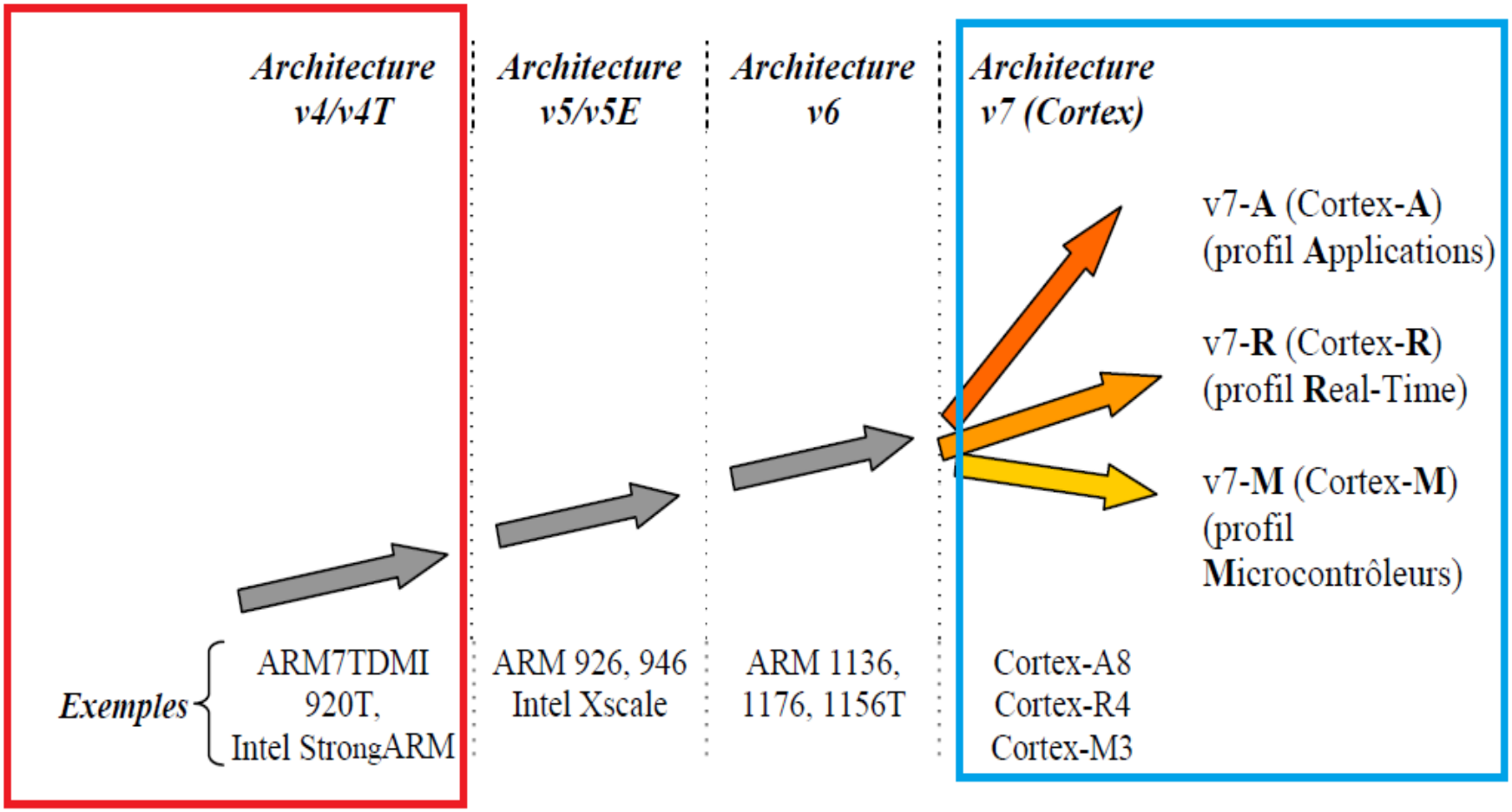
Adresse	Vecteur
0x00	Adresse de début de la pile principale
0x04	Reset
0x08	NMI
0x0C	Hard fault
0x10	MemManage Fault
0x14	Bus Fault
0x18	Usage Fault
0x1C – 0x2B	
0x2C	SVCall
0x30	Debug Monitor
0x34	Réservé
0x38	PendSV
0x3C	SYSTICK
0x40 ……	IRQ0 ……

Cortex-M3: Déroulement des Interruptions

Pour répondre à une interruption, le Cortex M3:

- Sauvgarde automatique de l'état du processeur dans la pile.
- Charge l'adresse de la routine d'interruption à partir de la table des vecteurs.
- Exécute la routine d'interruption.
- Restitue automatique de l'état précédent de la machine pour revenir au programme de départ.

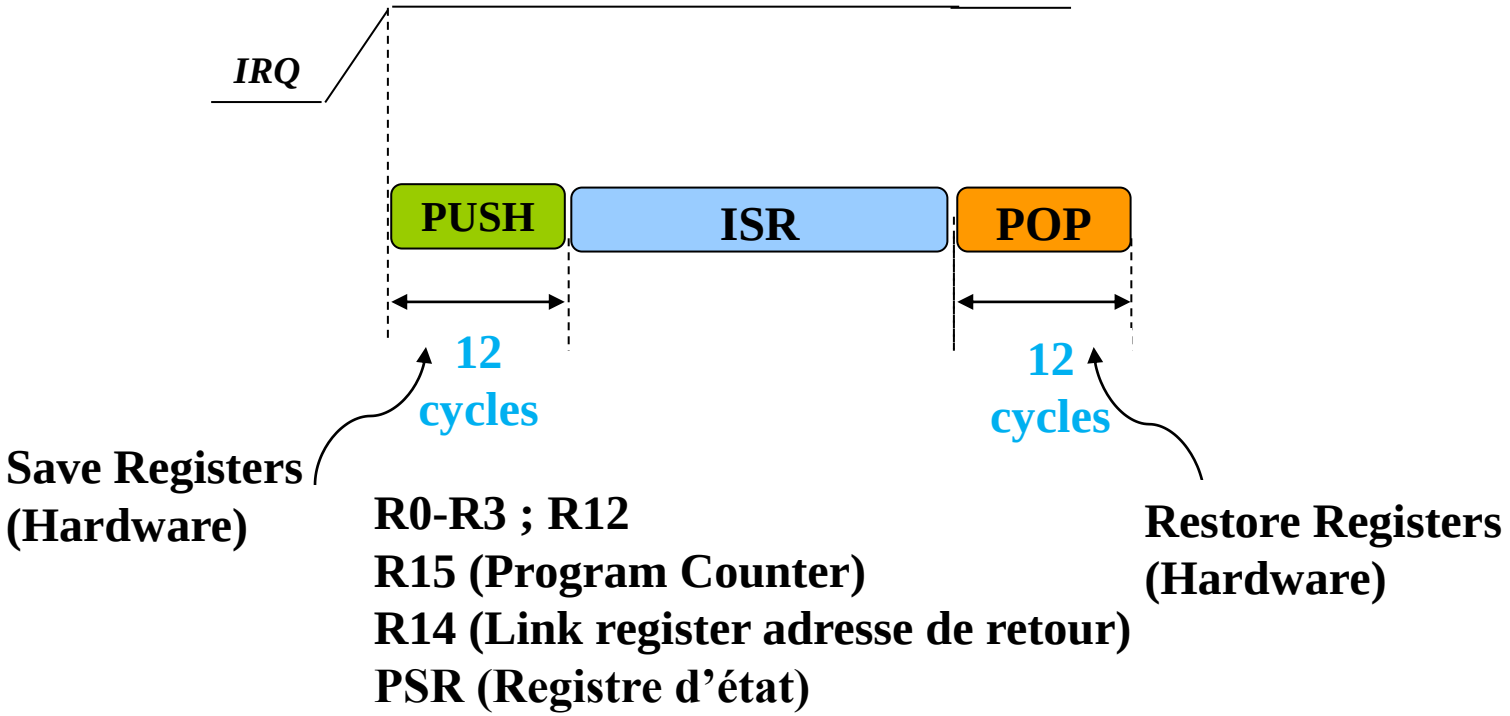
Cortex-M3: Déroulement des Interruptions



Evolution des architectures des cœurs ARM

Cortex-M3: Déroulement des Interruptions

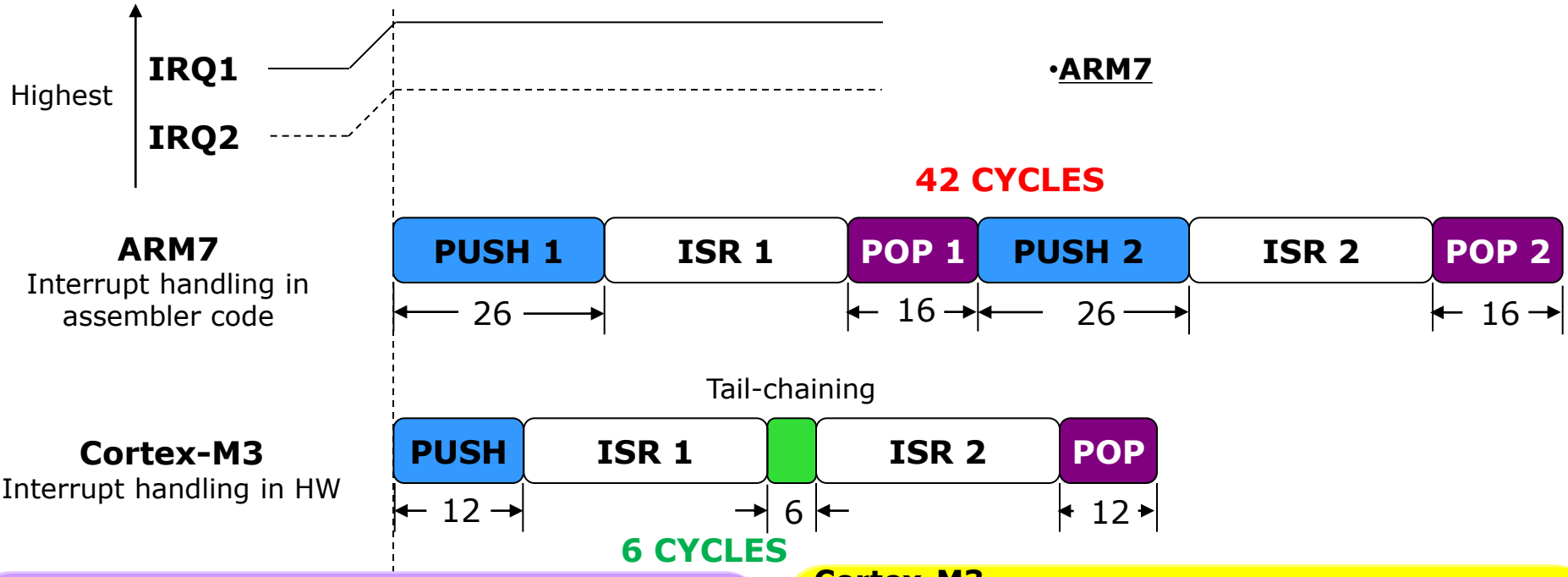
Temps de réponse d'une interruption Cortex-M3



Pour L'architecture du **ARM7** l'opérations PUSH nécessite 26 cycles et le POP nécessite 16 cycles d'horloges.

Cortex-M3: Déroulement des Interruptions

Tail Chaining: Recherche des Interruptions en attente



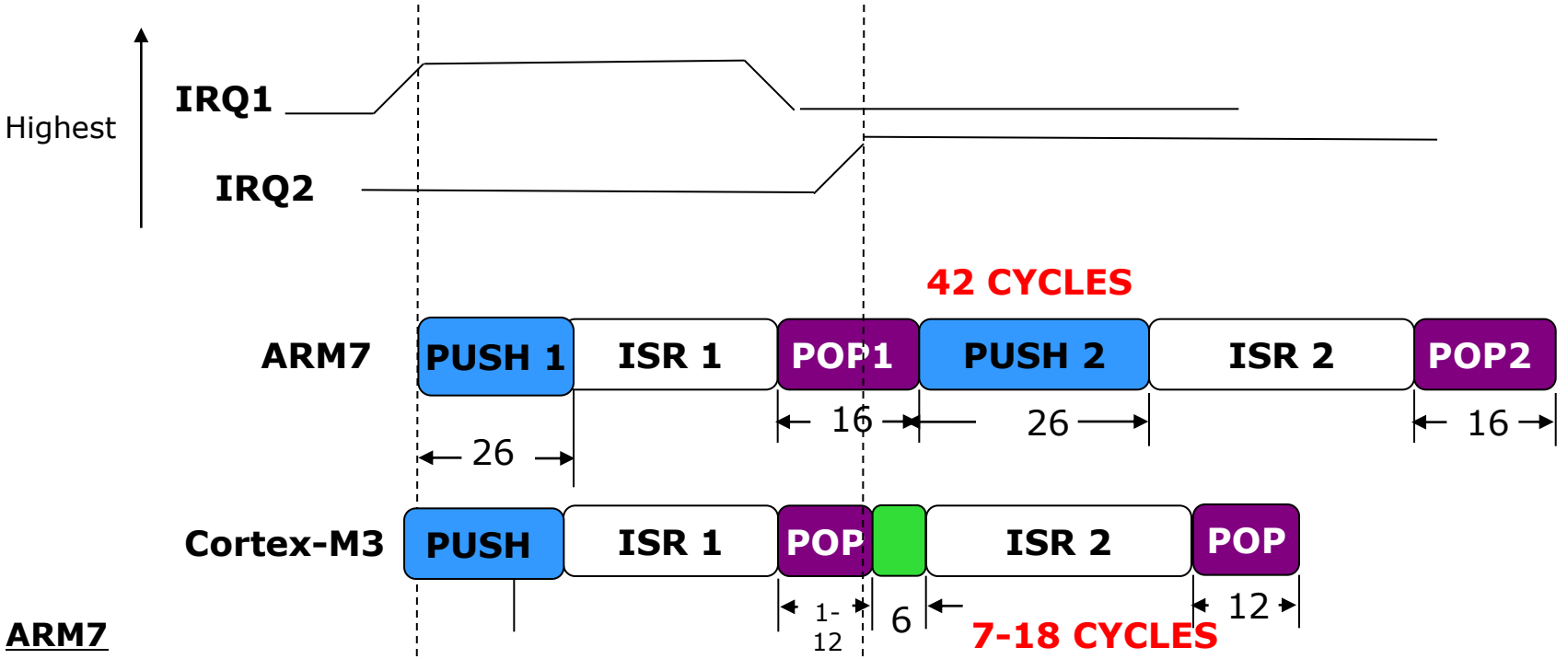
•**ARM7**
- 26 cycles pour déclencher l'exécution de ISR1
- 42 cycles pour passer de ISR1à ISR2
- 16 cycles pour retourner au prog principal

Cortex-M3

- 12 cycles pour déclencher l'exécution de ISR1
- 6 cycles pour passer de ISR1à ISR2
- 12 cycles pour retourner au prog principal

Cortex-M3: Déroulement des Interruptions

Preemption: priorité



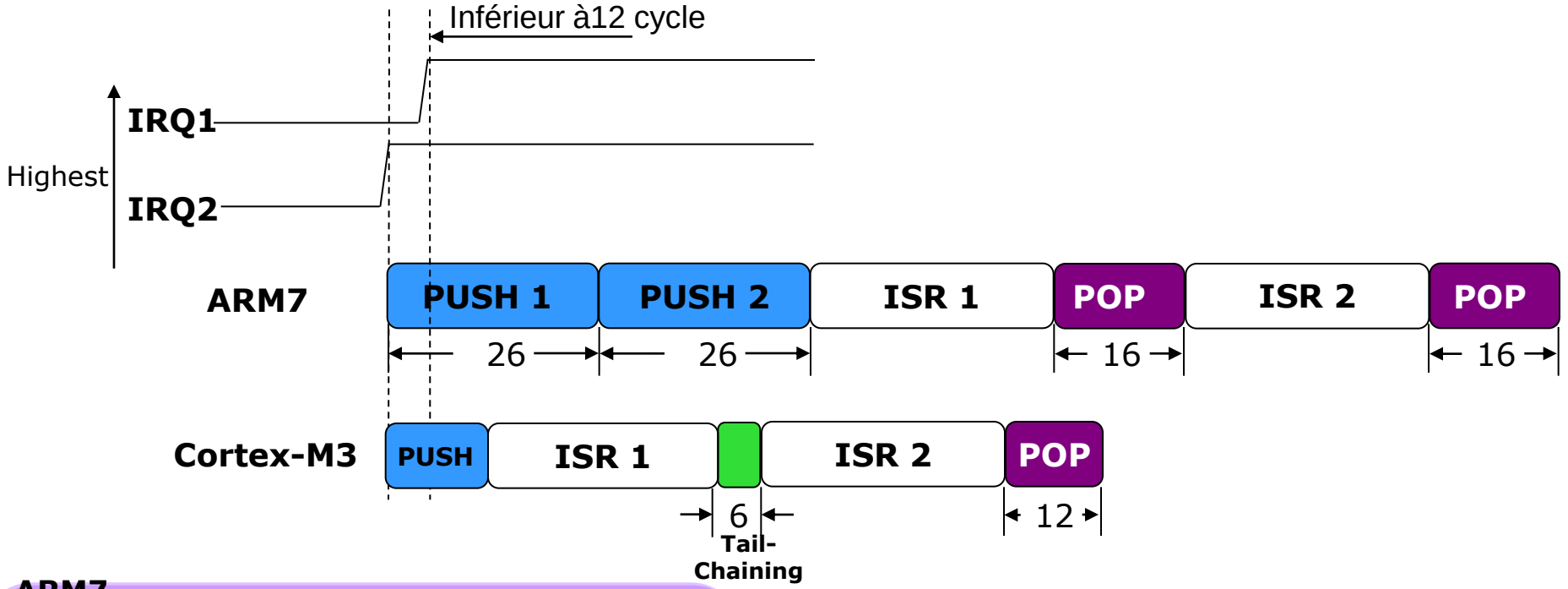
- Il sauvgarde le contexte du Prog (PUSH1)
- 26 cycles pour déclencher l'exécution de ISR1
- Récupérer le contexte (prog princi) (POP1)
- Pendant la récupération si une interruption se présente le pop termine sa tâche.
- 42 cycles pour passer de ISR1 à ISR2

Cortex-M3

- Pendant la récupération (POP) si une interruption se présente
- il abandonne le pop
- Il lance le tail chaining puis il déclenche l'exécution de ISR2
- 7 à 18 cycles pour passer de ISR1 à ISR2

Cortex-M3: Interrupt Handling

Late Arrival: Arrivé en retard de l'interruption de priorité



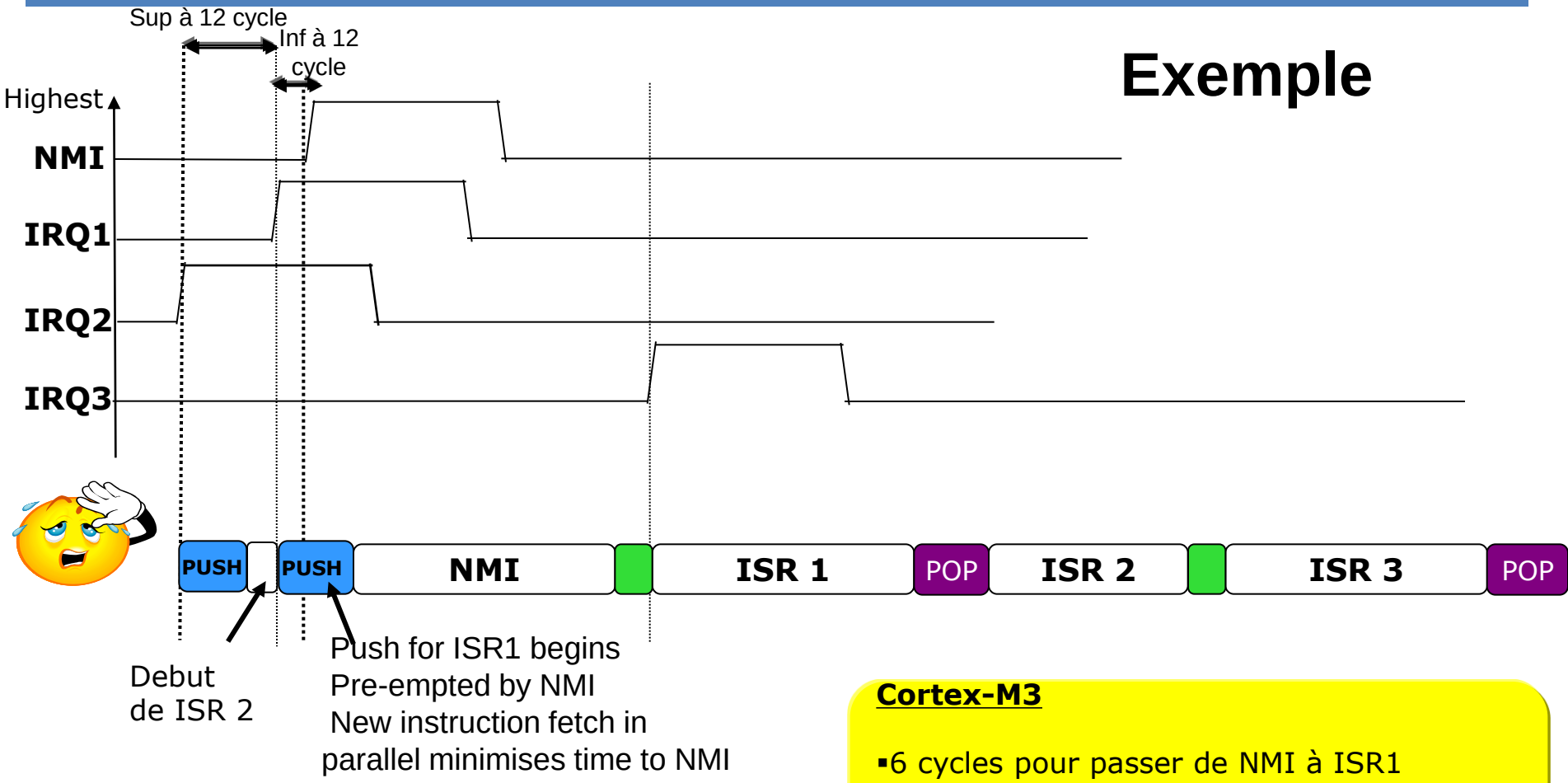
ARM7

- Sauvgarde le contexte du Prog(PUSH1)
- Sauvgarde de **ISR 2** (PUSH2)
- 52 cycles pour déclencher l'éxecution de **ISR 1**
- Récupérer le contenue des registres de **ISR 2**
- 16 cycles pour passer de **ISR 1 à ISR 2**

Cortex-M3

- Sauvgarde le contexte du Prog (PUSH)
- 12cycles pour déclencher l'éxecution de **ISR 1**
- 6cycles pour passer de **ISR1** à **ISR 2**

Cortex-M3: Interrupt Handling



Exemple

Cortex-M3

- 6 cycles pour passer de NMI à ISR1
- 12 cycles pour passer de ISR1 à ISR2
- 6 cycles pour passer de ISR2 à ISR3

Cortex-M3: Gestion d'alimentation

SLEEP MODE (SLEEP NOW et SLEEP ON EXIT)

Sleep-now mode	Description
Mode en entrée (Sommeil) Faible consommation	WFI (Wait for Interrupt) ou WFE (Wait for Event) tant que : – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Dans le registre de contrôle du système Il entre immédiatement en mode sommeil
Mode en sortie(réveil)	Si WFI est utilisée comme entrée: Il se réveil avec une interruption Si WFE est utilisée comme entrée: Il se réveil avec un événement

Cortex-M3: Gestion d'alimentation

SLEEP MODE (SLEEP NOW et SLEEP ON EXIT)

Sleep-on-exit mode	Description
Mode en entrée (Sommeil) Faible consommation	WFI (wait for interrupt) tant que: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Il termine la routine d'interruption et entre en mode sommeil (sleep)
Mode en sortie(réveil)	Il se réveil avec une interruption

Cortex-M3: Gestion d'alimentation

- STOP MODE (DEEP SLEEP)

STOP MODE	Description
Mode d'entrée	WFI (Wait for Interrupt) ou WFE (Wait for Event) tant que: –bit SLEEPDEEP =1 de registre de controle (CSR) – bit PDDS (Power down deepsleep.) =0 de registre de controlde de consommation (PWR_CR)
Mode de sortie	Si WFI est utilisée comme entrée: Certaines lignes EXTI sont configurées en mode interruption. Si WFE est utilisée comme entrée: Certaines lignes EXTI sont configurées en mode événement

Cortex-M3: Gestion d'alimentation

STANDBY MODE

STANDBY MODE	Description
Mode d'entrée	WFI (Wait for Interrupt) ou WFE (Wait for Event) tan que: – bit SLEEPDEEP =1 – bit PDDS = 1 – bit WUF (wakeup flag)=0 dans le registre Power Control/Status (PWR_CSR)
Mode de sortie	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.

Cortex-M3: SysTick (System Timer)

- Flexible system timer
- Le Cortex ™ M3 dispose d'un timer intégré dans le noyau appelé l'horloge du système (SysTick).
- Ceci permet de générer une interruption pour un système d'exploitation. C'est un décompteur 24-bit rechargeable automatiquement avec la génération d'interruption à la fin du comptage.

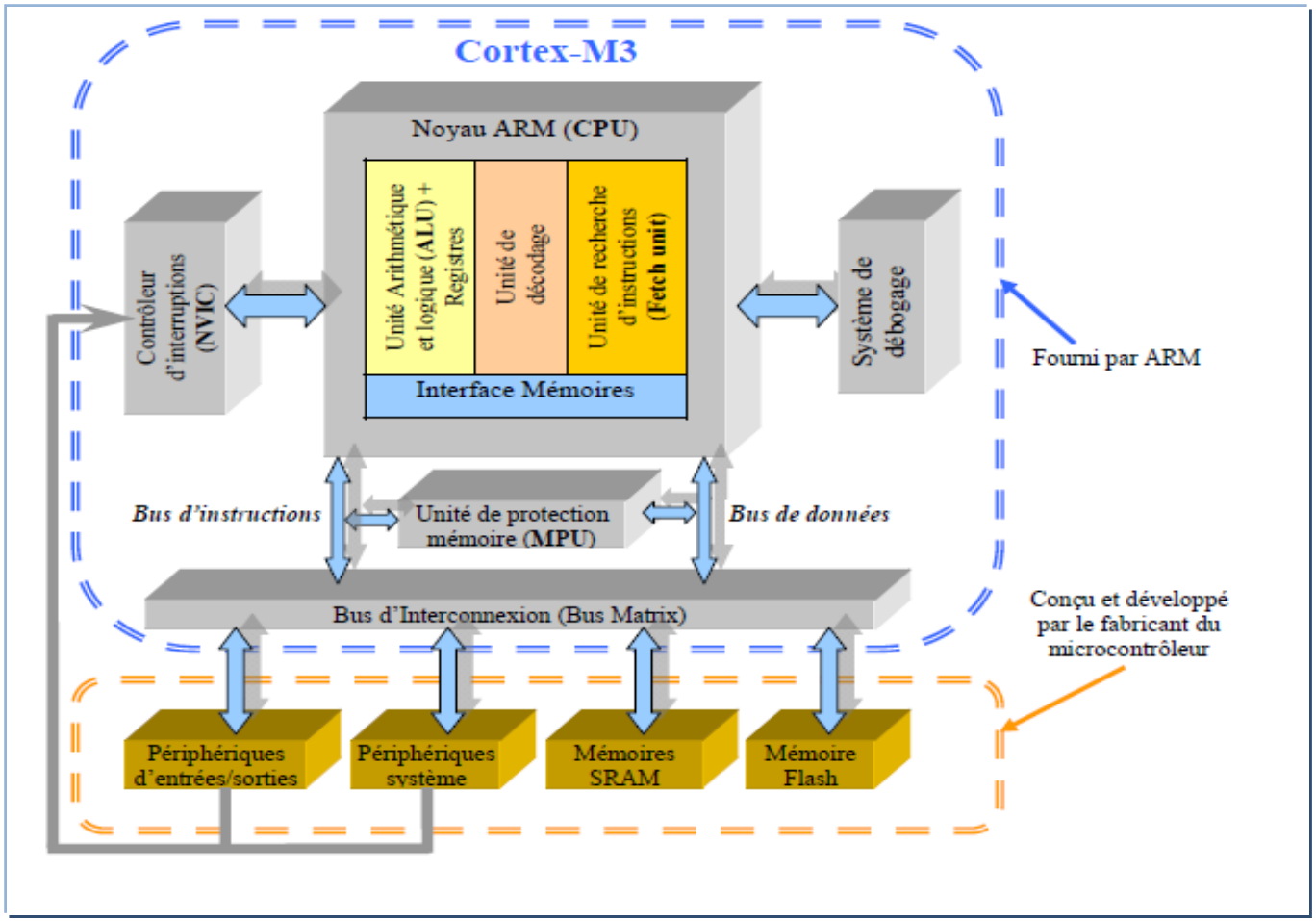
  Adapte pour les systèmes à temps réel ou autres taches planifiées.

- **PourSTM32F10x l'horloge SysTick peut être: l'horloge du processeur CPU ou une horloge CPU / 8**

Outline

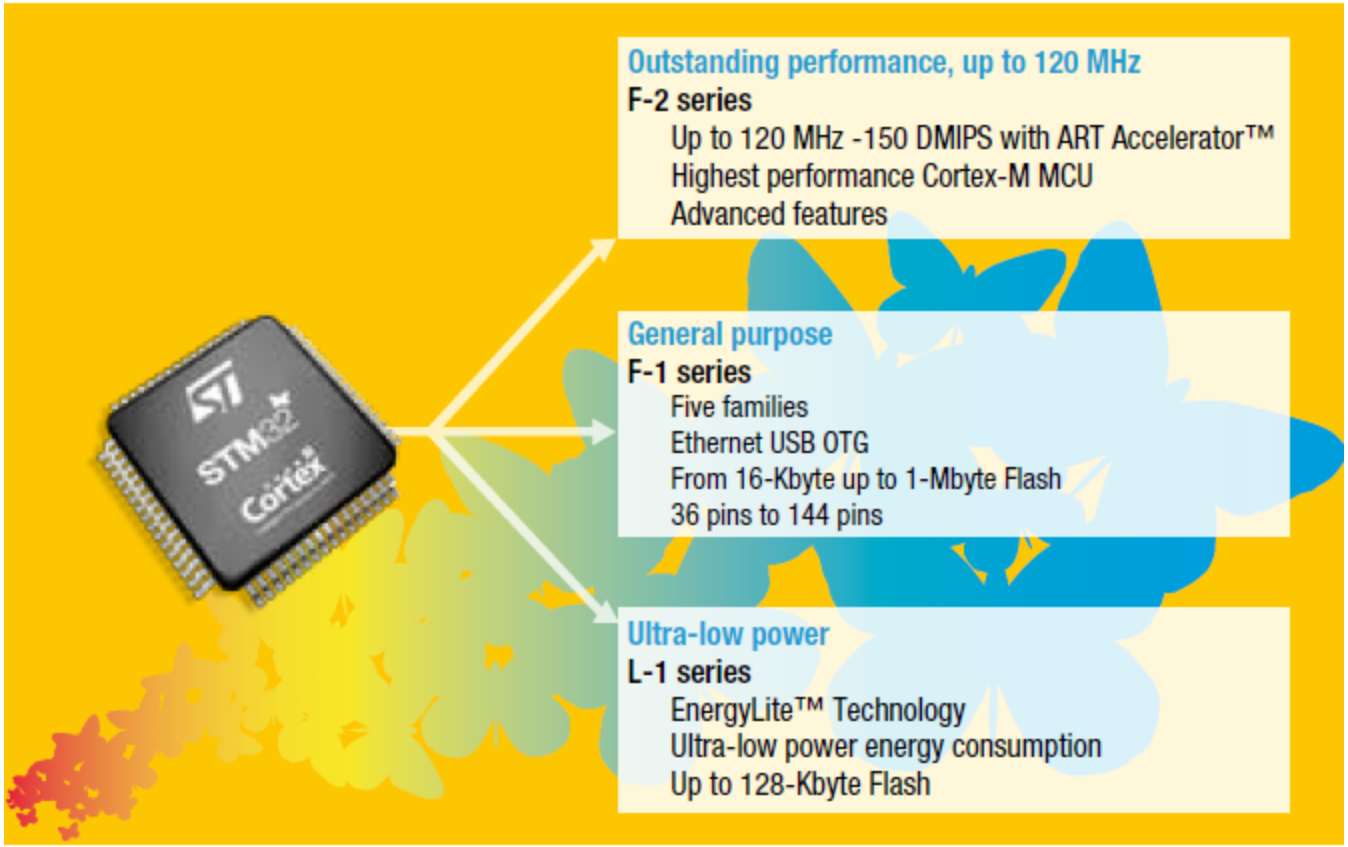
- Introduction
- Cortex-M3
- Cortex-M3 based MCUs: The STM32F10x Family
- STM32 Programming
- STM32VL Discovery Kit

Le Microcontroleur



Architecture générique d'un microcontrôleur à base du coeur Cortex-M3

STM32 Series



STM32: The Cortex-M3 MCU F1 Family

•Designation: STM32F***10x y z:***

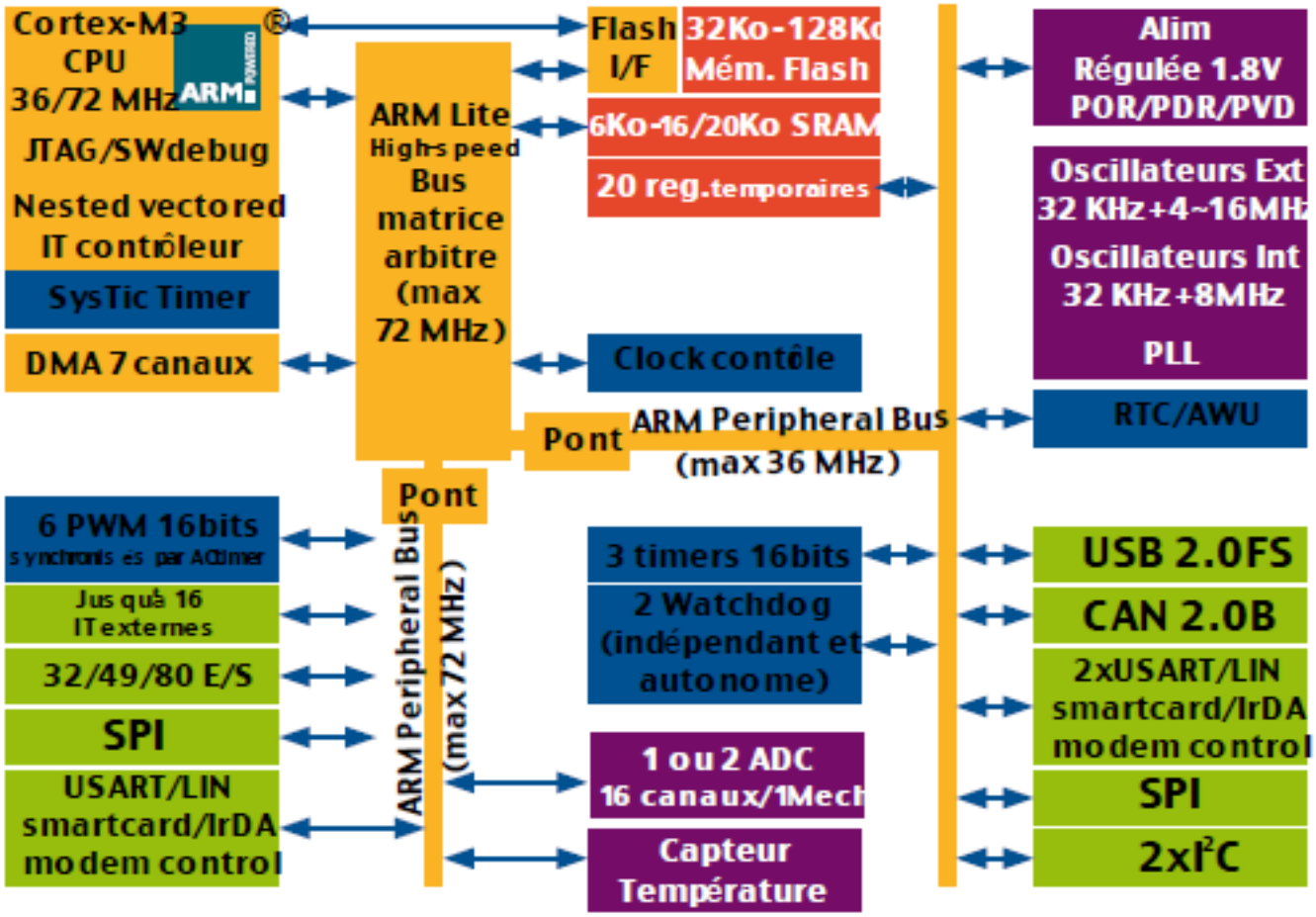
10x : désigne la catégorie (ou ligne de produit) à laquelle appartient le µc

y: (y est une référence qui désigne le nombre de pins ainsi que le type de package du circuit intégré (T: 36 pins, C: 48 pins, R : 64 pins, V : 100 pins et Z : 144 pins).

z: donne une indication sur la taille de la mémoire flash interne (4 pour 16 Kbytes, 6 pour 32 Kbytes, 8 pour 64 Kbytes, B pour 128 Kbytes, C pour 256 Kbytes, D pour 384 Kbytes et E pour 512 Kbytes).

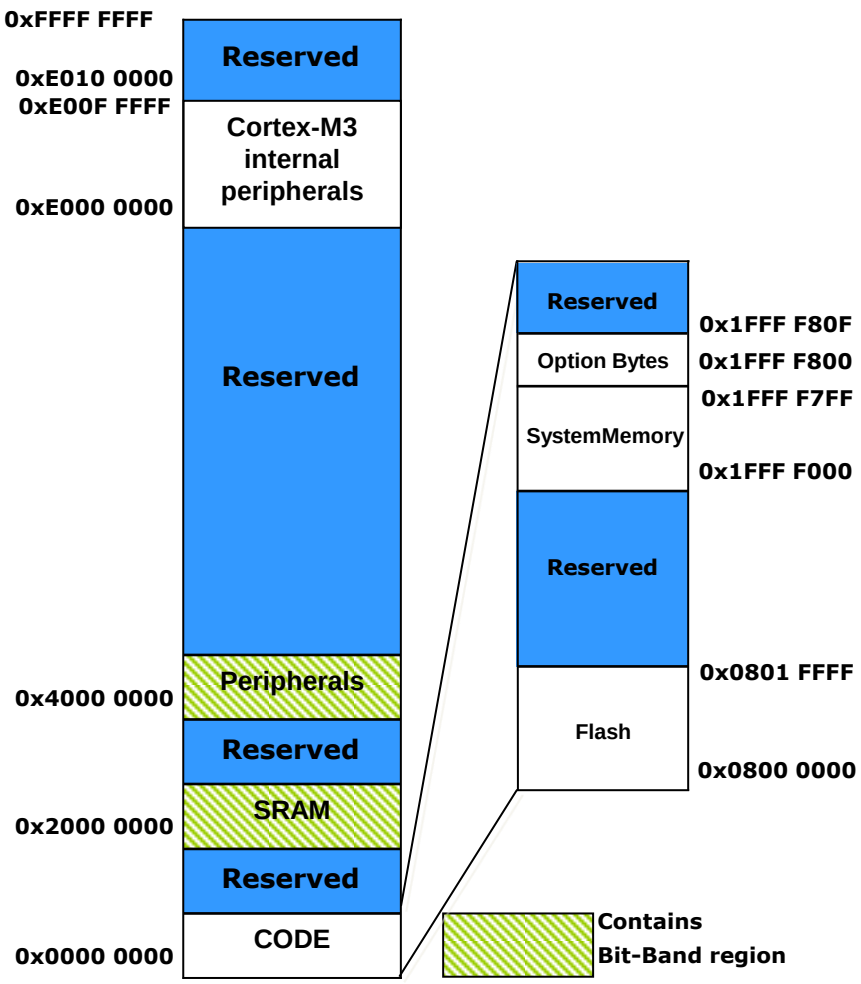
Ligne de produit	*Référence 10x*	*Composants communs*	*Fréquence CPU*	*Mémoire SRAM*	*Périphériques spécifiques*
Performance	103	• USART (jusqu'à 5) • DAC 12 bits (jusqu'à 2) • Timers 16 bts (jusqu'à 6) • Canaux DMA (jusqu'à 12) • Oscillateurs RC internes (40 KHz et 8 MHz) • Horloge temps réel. •	72 MHz	64 Kbytes	USB, SDIO, CAN, I^2S et Timer PWM.
Accès USB (USB Access)	102		48 MHz	16 Kbytes	USB
Accès (Access)	101		36 MHz	48 Kbytes	
Connectivité (Connectivity)	105 & 107		72 MHz	Jusqu'à 64 Kbytes	USB, CAN, I^2S (classe audio) et Ethernet
VALUE	**100**		**24MHz**	**de 32 à 512 Kbytes**	

STM32F10x Series Block Diagram



DMA : Direct Memory Access
RTC : Real Time Clock
AWU : Auto Wake Up capability with RTC alarm
POR : Power On Reset
PDR : Power Down Reset
PVD : Programmable Voltage Dete

STM32F10x: Memory Mapping and Boot Modes



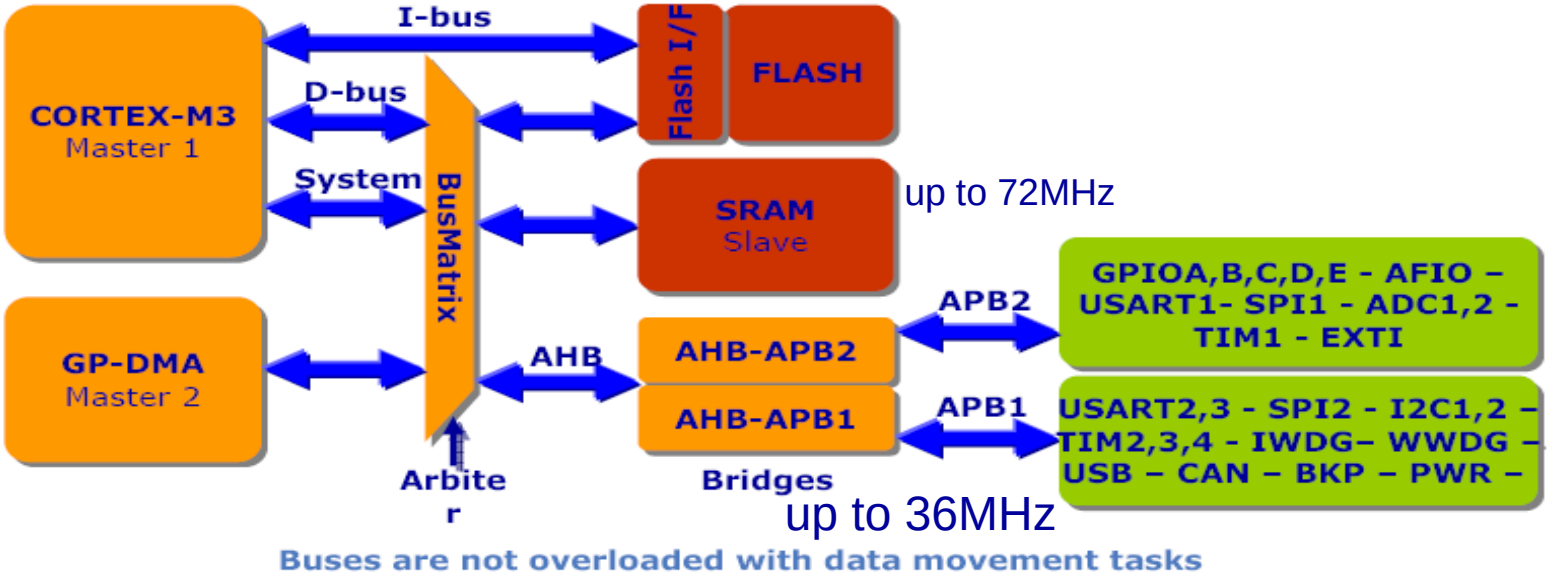
- Addressable memory space of 4 GBytes
- RAM : up to 96 kBytes
- FLASH : up to 1MBytes

- Boot modes:
  - Depending on the Boot configuration
    - Embedded Flash Memory
    - System Memory
    - Embedded SRAM Memory

BOOT Mode Selection Pins		**Boot Mode**	**Aliasing**
BOOT1	BOOT0		
x	0	User Flash	User Flash is selected as boot space
0	1	SystemMemory	SystemMemory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

System architecture

• ***I-Bus*** (Instruction Bus): dédié principalement pour le transfert des instructions (code) entre Cortex et la mémoire Flash.
• ***D-Bus*** & ***System Bus***: connectés directement à la matrice de bus qui sert de lien avec les bus d'accès à la SRAM, le système DMA (Direct Memory Access) ainsi que le **AHB** (Advnaced High Bus) .
• ***APB1*** & **2** (Advanced Peripheral Bus 1 & 2): Chacun des deux bus est connecté au AHB travers un pont (respectivement AHB-APB1 et AHB-APB2) et servent à véhiculer les données vers et depuis une partie des périphériques.



Définition : L'interface parallèle GPIO (General Purpose Input Output)

• Ce type d'interface, répartie sur plusieurs ports (maximum 8 bits), permet de prendre en compte des états logiques appliqués en entrée (état de capteurs) ou de générer des signaux binaires en sortie (commande d'actionneurs).

• Les broches de ces ports peuvent donc être configurées en entrée ou en sortie, avec différentes options (résistances de rappel, sorties collecteurs ouverts, interruption...).

• La configuration ainsi que l'état logique de ces broches est obtenue par des opérations d'écriture ou de lecture dans différents registres associés à chaque port.
• On trouve généralement :

- Un registre de direction pour une configuration en entrée ou en sortie,
- Un registre de donnée recopiant les états logiques de chaque broche de port,
- Un registre d'option permettant plusieurs configurations en entrée ou en sortie.

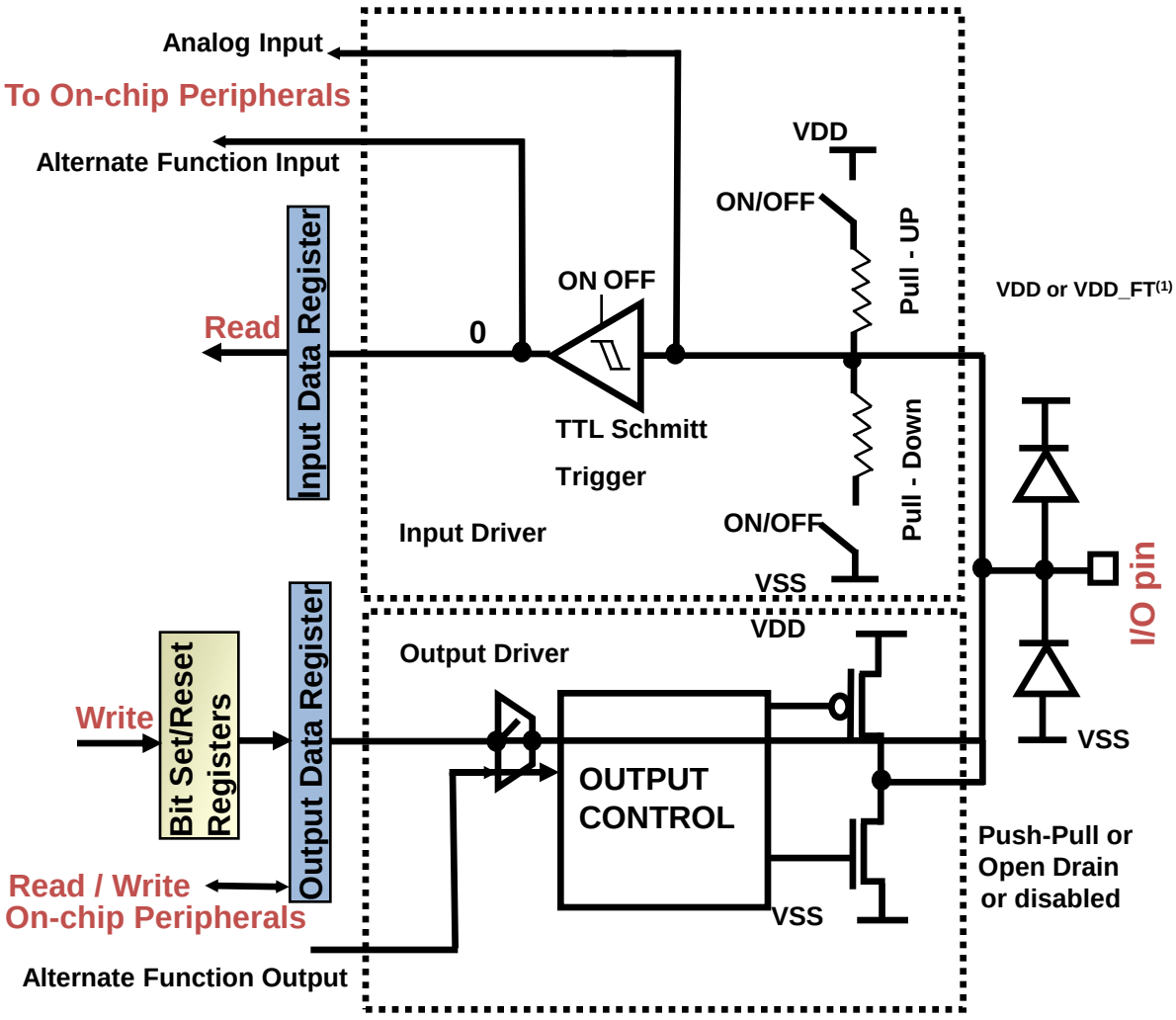
General Purpose Input Output (GPIO)

- Tous les I/O sont réparties sur 5 ports (GPIOA…GPIOE)
- Chaque general purpose I/O port possède:
  - 2 registres (32 bits) de configurations
  - 2 registres (32 bits) de données
  - 1 registre (32 bits) set/reset
  - 1 registre (16bits) reset
  - 1 registre (32bits) locking registre
- Chaque port bit d'un GPIO peut être configuré en un de ces mode (Input floating, Input pull-up,Input-pull-down,Analog,Output open-drain,Output push-pull, Alternate function push-pull,Alternate function open-drain
- Alternate Functions pins (like USARTx, TIMx, I2Cx, SPIx, CAN, USB…)
- Configurable Output Speed up to 50 MHz

GPIO Configuration Modes

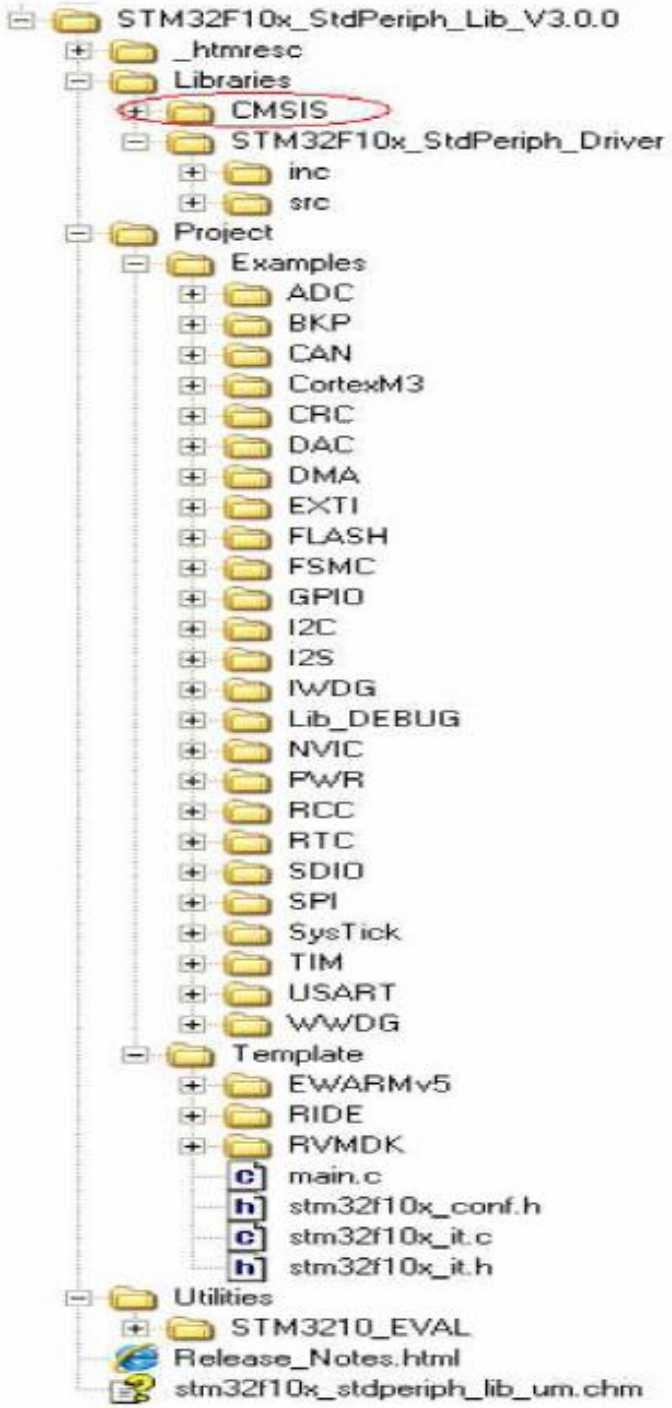
Configuration Mode	CNF1	CNF0	MOD1	MOD0
Analog Input	0	0	00	
Input Floating **(Reset State)**	0	1		
Input Pull-Up(2)	1	0		
Input Pull-Down(2)				
Output Push-Pull	0	0	01: 10 MHz 10: 2 MHz 11: 50 MHz	
Output Open-Drain	0	1		
AF Push-Pull	1	0		
AF Open-Drain	1	1		

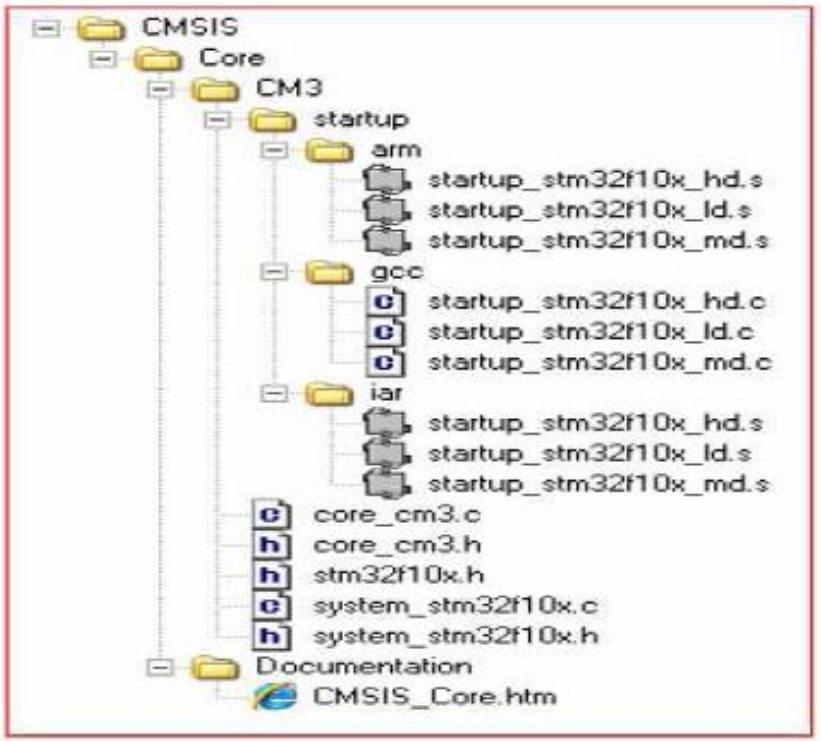
(2) ***Input Pull-Up and Input Pull-Down are differentiated by the PxODR.y bit field.***



Outline

- Introduction
- Cortex-M3
- Cortex-M3 based MCUs: The STM32F10x Family
- STM32 Programming
- STM32VL Discovery Kit


STM32F10x_StdPeriph_Lib_V3.0.0
_htmresc
Libraries
CMSIS
STM32F10x_StdPeriph_Driver
inc
src
Project
Examples
ADC
BKP
CAN
CortexM3
CRC
DAC
DMA
EXTI
FLASH
FSMC
GPIO
I2C
I2S
IWDG
Lib_DEBUG
NVIC
PWR
RCC
RTC
SDIO
SPI
SysTick
TIM
USART
WWDG
Template
EWARMv5
RIDE
RVMDK
main.c
stm32f10x_conf.h
stm32f10x_it.c
stm32f10x_it.h
Utilities
STM3210_EVAL
Release_Notes.html
stm32f10x_stdperiph_lib_um.chm


CMSIS
Core
CM3
startup
arm
startup_stm32f10x_hd.s
startup_stm32f10x_ld.s
startup_stm32f10x_md.s
gcc
startup_stm32f10x_hd.c
startup_stm32f10x_ld.c
startup_stm32f10x_md.c
iar
startup_stm32f10x_hd.s
startup_stm32f10x_ld.s
startup_stm32f10x_md.s
core_cm3.c
core_cm3.h
stm32f10x.h
system_stm32f10x.c
system_stm32f10x.h
Documentation
CMSIS_Core.htm

-*Liste des interruptions du STM32 spécifiques au noyau Cortex-M3.*
-*Organisation de la mémoire STM32 et définition des adresses physiques des registres.*
- *Options de configuration.*

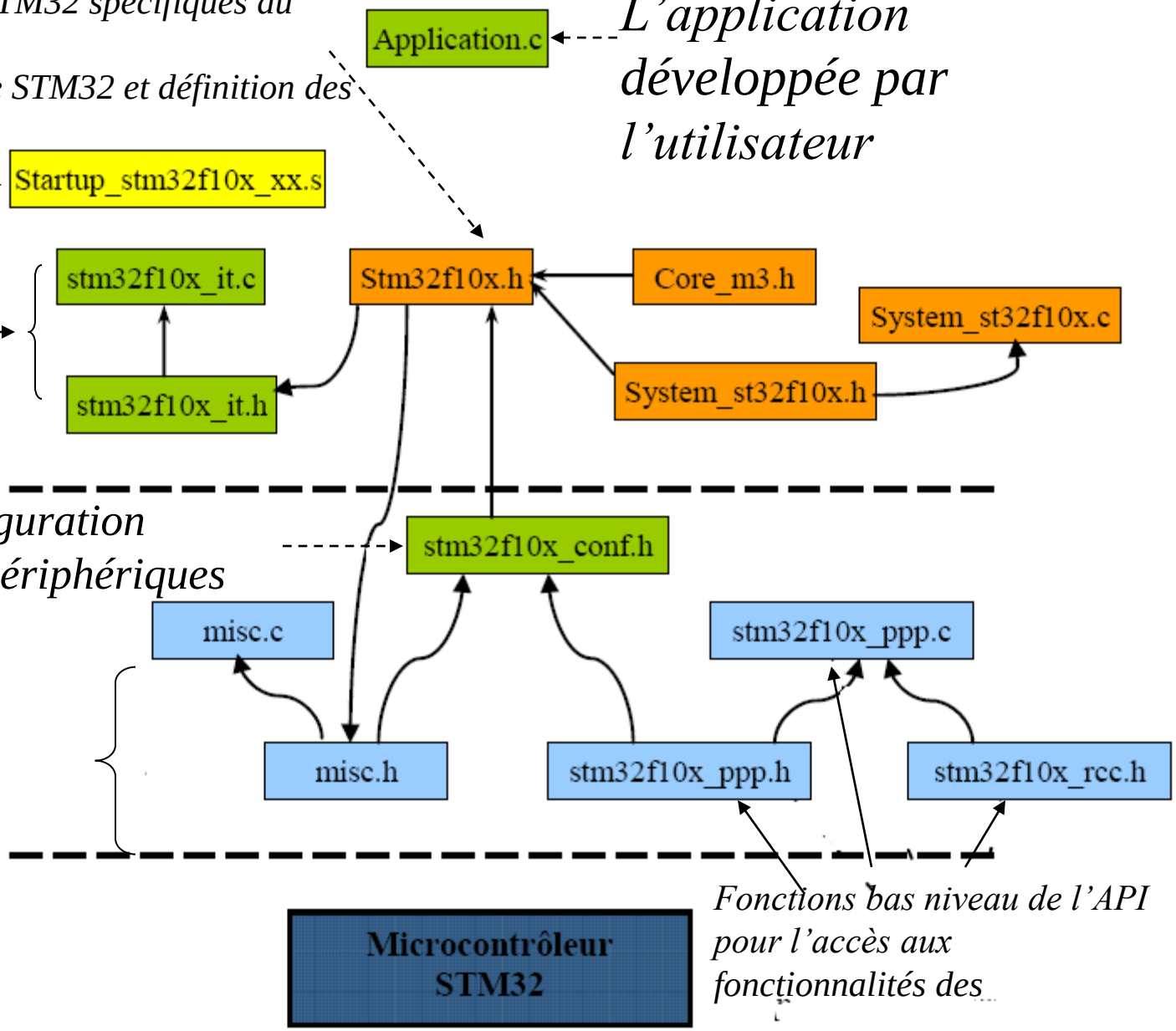
Fichier de démarrage (assembleur) pour l'initialisation des paramètres du STM32

Routines d'interruptions exceptions du Cortex-M3

L'initialisation de l'utilisateur

Fichier entête de configuration contenant la liste des périphériques utilisés du STM32

Les drivers des périphériques de base (noyau) (Contrôleur d'interruptions, Timer système)



L'application développée par l'utilisateur

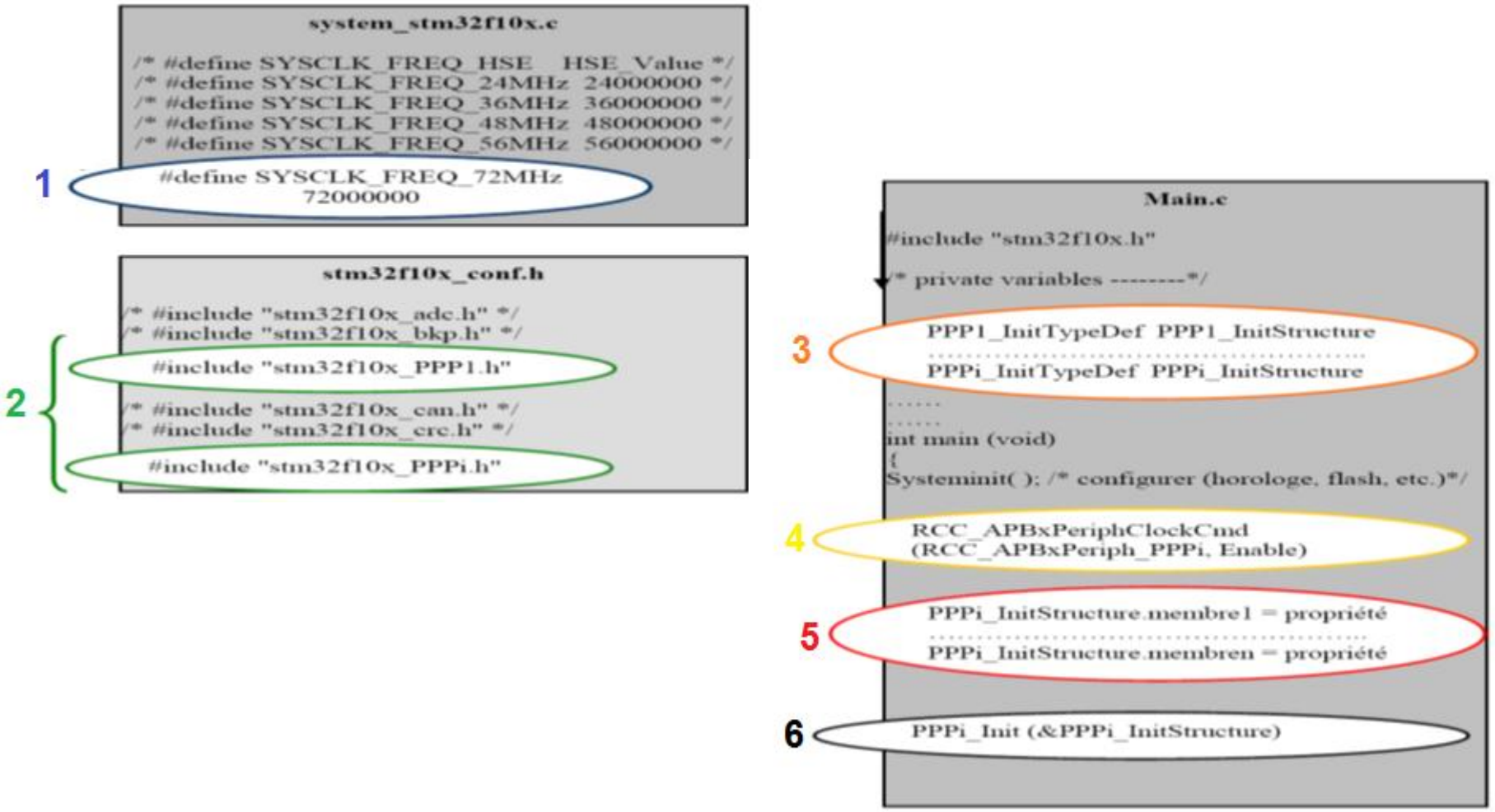
Fonctions bas niveau de l'API pour l'accès aux fonctionnalités des périphériques

STM32 Programming

- stm32f10x_ppp.c et .h n'est pas uniquement utilisé pour désigner un seul fichier mais elle s'étend à un certain nombre de fichiers.
- Chaque fichier étant relatif à l'un des périphériques du microcontrôleur (ppp étant remplacée par l'abréviation du périphérique).

Nom de fichier	*Périphérique*
stm32f10x_adc	Convertisseur Analogique/Numérique
stm32f10x_bkp	Sauvegarde de données
stm32f10x_can	Bus CAN
stm32f10x_crc	Calcul des codes de détection d'erreurs
stm32f10x_dac	Convertisseur Analogique/Numérique
stm32f10x_dma	Accès direct à la mémoire
stm32f10x_exti	Interruptions externes
stm32f10x_gpio	Entrées/sorties parallèles
stm32f10x_iwdg	Chien de garde
stm32f10x_pwr	Gestion du mode d'alimentation
stm32f10x_rcc	Contrôle du reset et de la distribution du signal d'horloge
stm32f10x_rtc	Horloge temps réel
stm32f10x_tim	Les Timers (compteurs).
stm32f10x_usart	L'interface de communication série synchrone et asynchrone
misc	Fonctions haut niveau du contrôleur d'interruptions et du timer système (SysTick)

Etapes de Développement d'application



Now you are able to…

Develop your application around STM32F100 device