Introduction
organisation - STM32 ? - performance - marché

ARM-Cortex
cpu - adresses - données – interruptions

Outil Keil
cible – config – librairie

Périphériques
horloges - gpio - dma - adc – timers

Bus Can
Intérêt - protocole – erreurs – timing – stm32 - application

Se familiariser avec

- Architecture ARM 32 bits
- Les périphériques et leurs drivers
- Le C embarqué
- Le Debug

Métriques

- TP
- Test écrit
- Rattrapage écrit

- **Livre:** The definitive Guide to the ARM Cortex-M3 *de Joseph Yiu*

  Le Bus Can Description *de Dominique Paret*

- **Microcontrôleur STM32F103RB**

  Users'manual:
  http://www.st.com/stonline/products/literature/rm/13902.pdf

  Data Sheet:
  http://www.st.com/stonline/products/literature/ds/13587.pdf

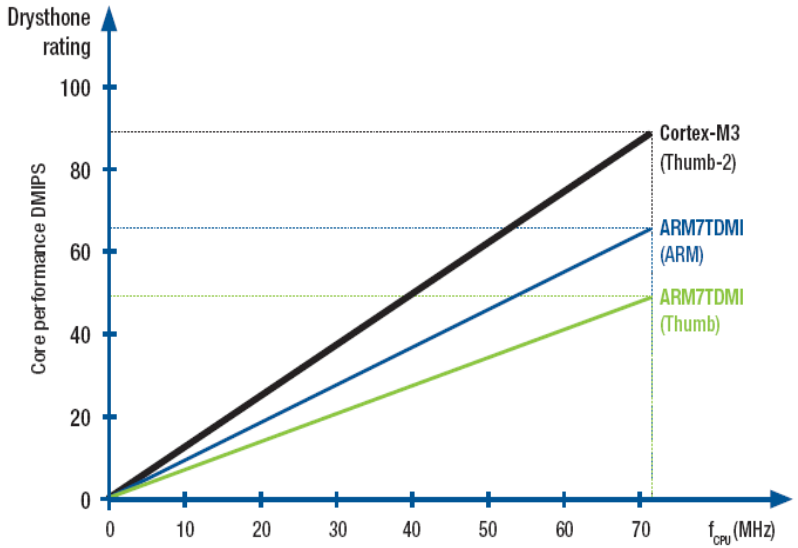
- **Kit MCBSTM32**

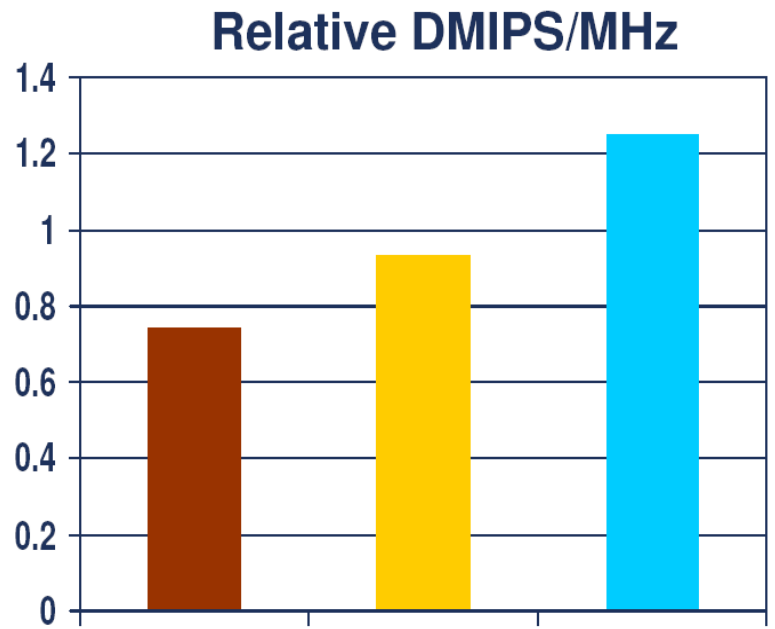
  Evaluation board :
  http://www.keil.com/arm/mcbstm32/

  User's guide :
  http://www.keil.com/support/man/docs/mcbstm32/

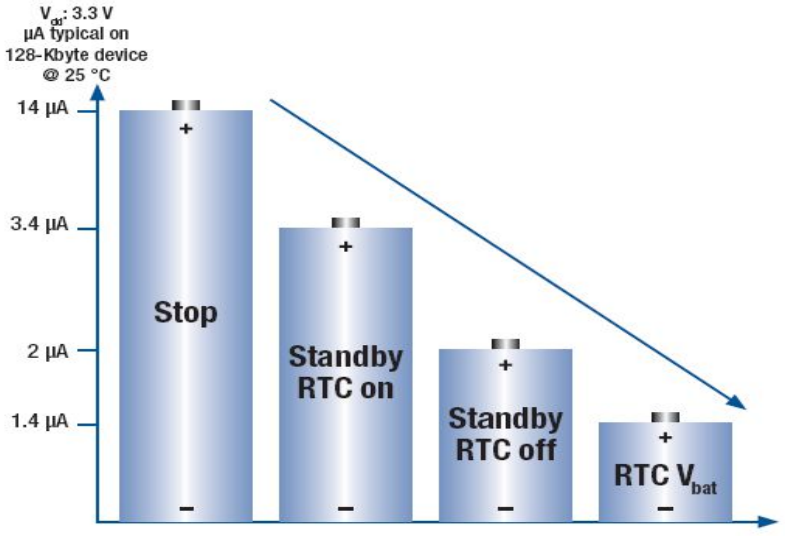
POURQUOI UN NOUVEAU ARM ?

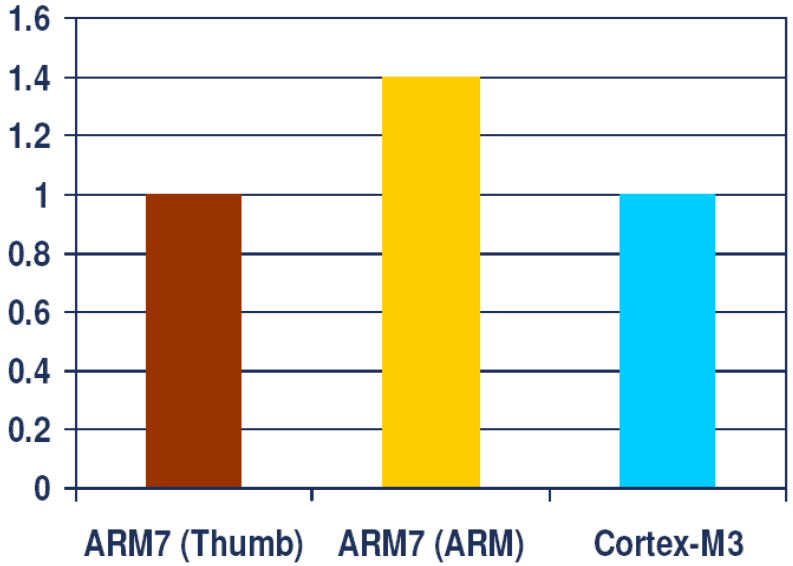
Performance





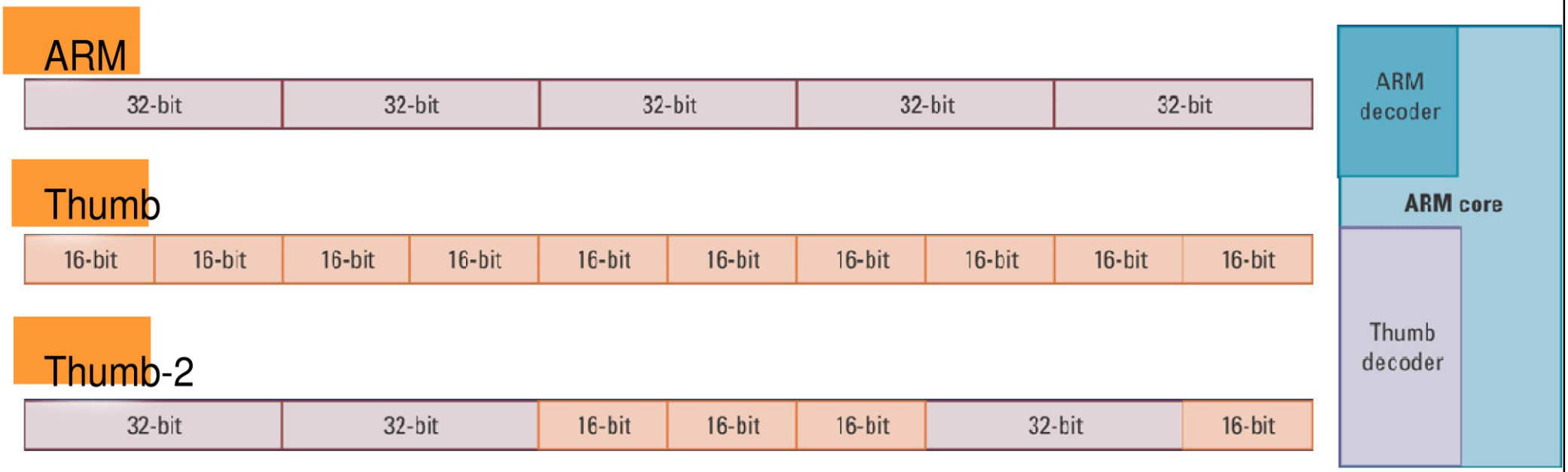
Consommation



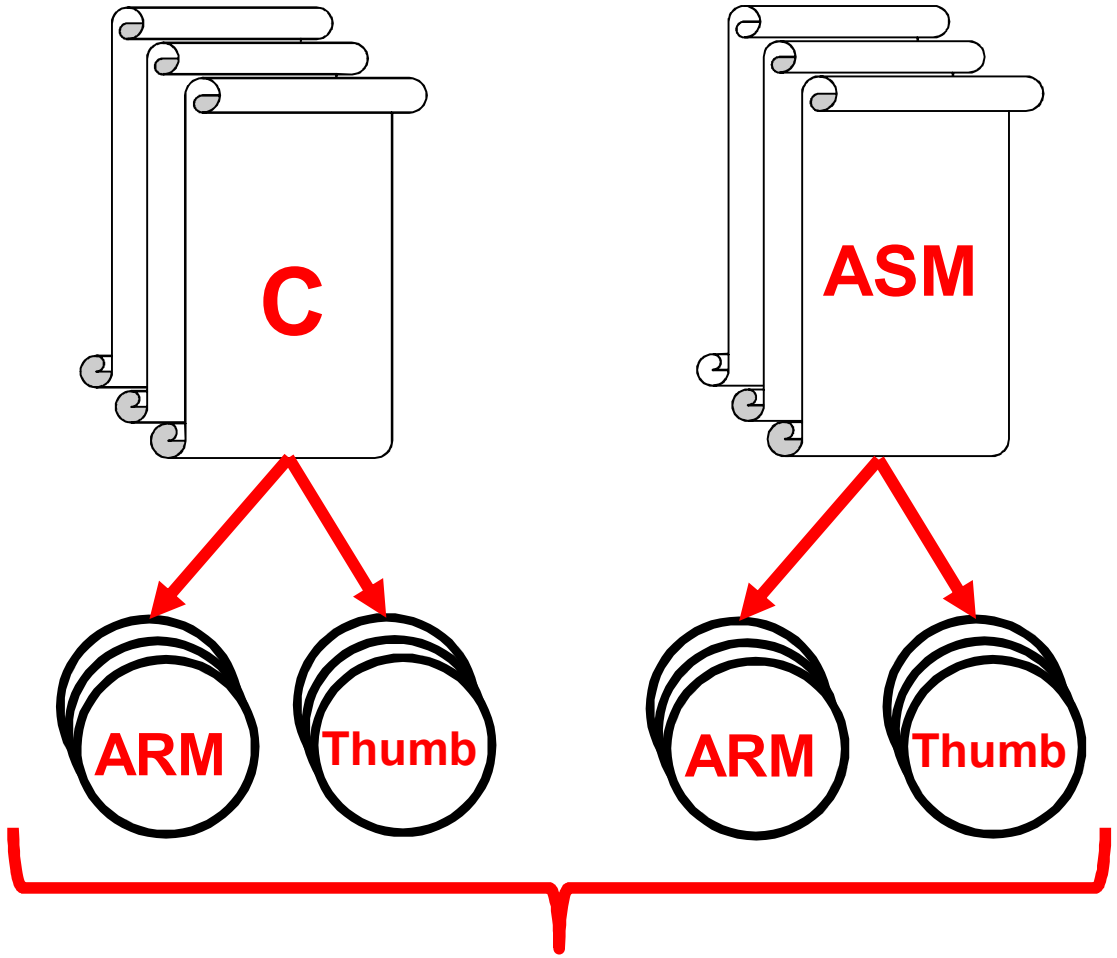


- 32 bit performance
- 16 bit code densité
- 1 seul jeux d'instruction

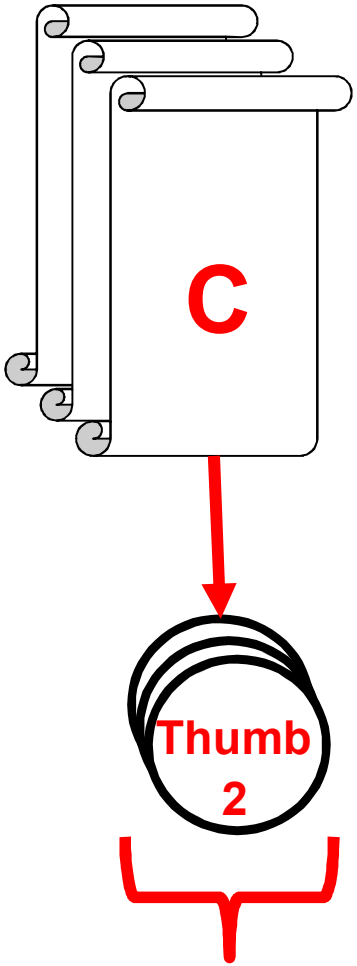
} THUMB-2



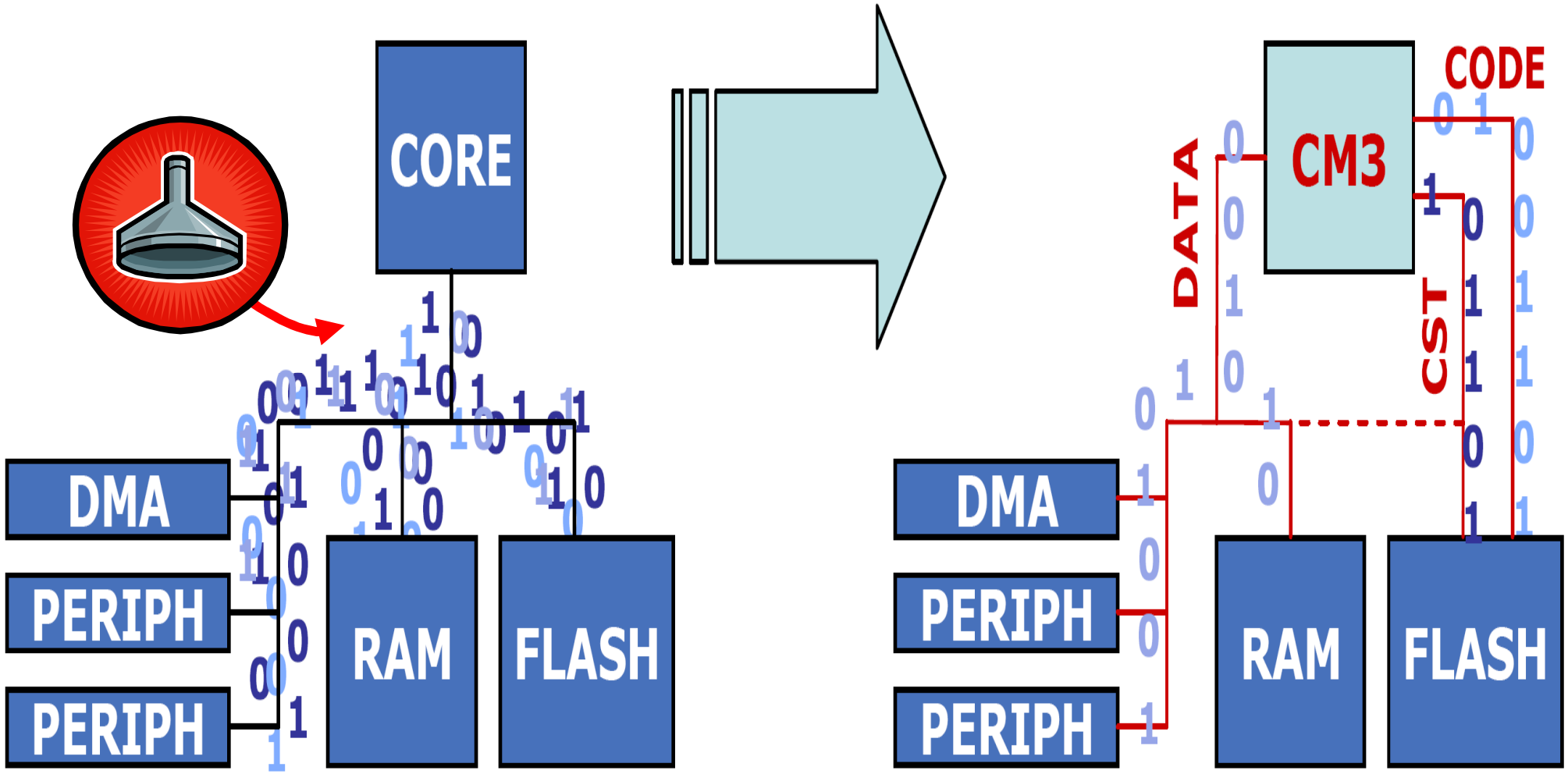
- ARM7 Objet



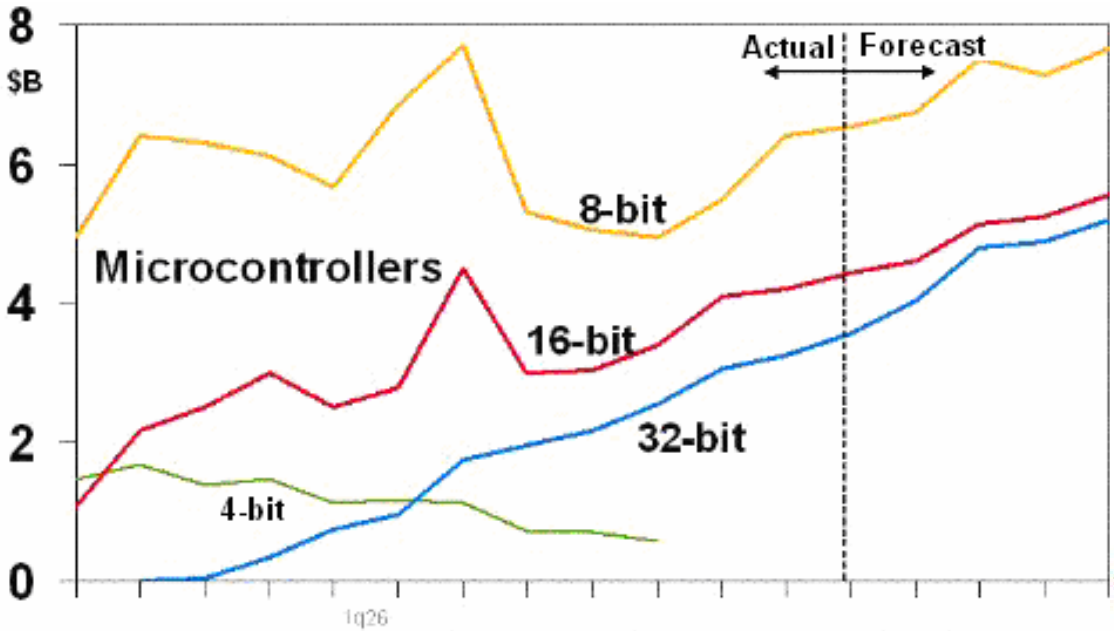
- Cortex Objet

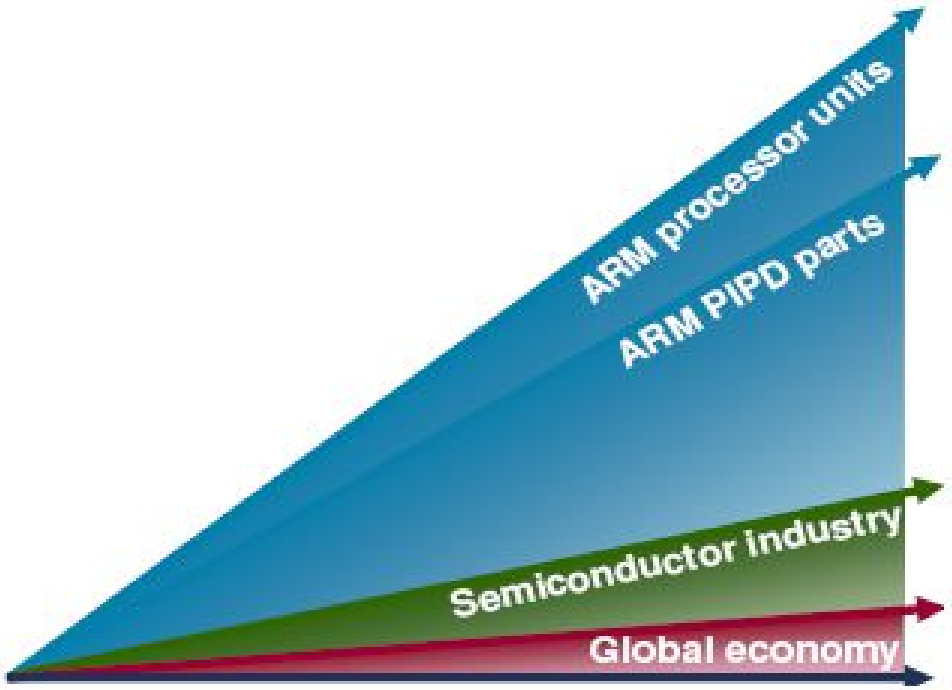


- Von Neumann
- Harvard



Croissance


8
$B
6
4
2
0
Actual
Forecast
8-bit
Microcontrollers
16-bit
32-bit
4-bit
1994 1996 1998 2000 2002 2004 2006 2008 2010
Source: Gartner Dataquest


ARM processor units
ARM PIPD parts
Semiconductor industry
Global economy

- Différences entre les jeux d'instruction ARM, Thumb et Thumb2 ?
- Différence entre l'architecture Von Neumann et Harvard ?
- Comment caractérise-t-on les performances d'un microcontrôleur ?
- Pourquoi la consommation est si importante sur un microcontrôleur ?
- Pourquoi le Cortex est plus simple à programmer que les anciennes familles ARM ?

SOMMAIRE

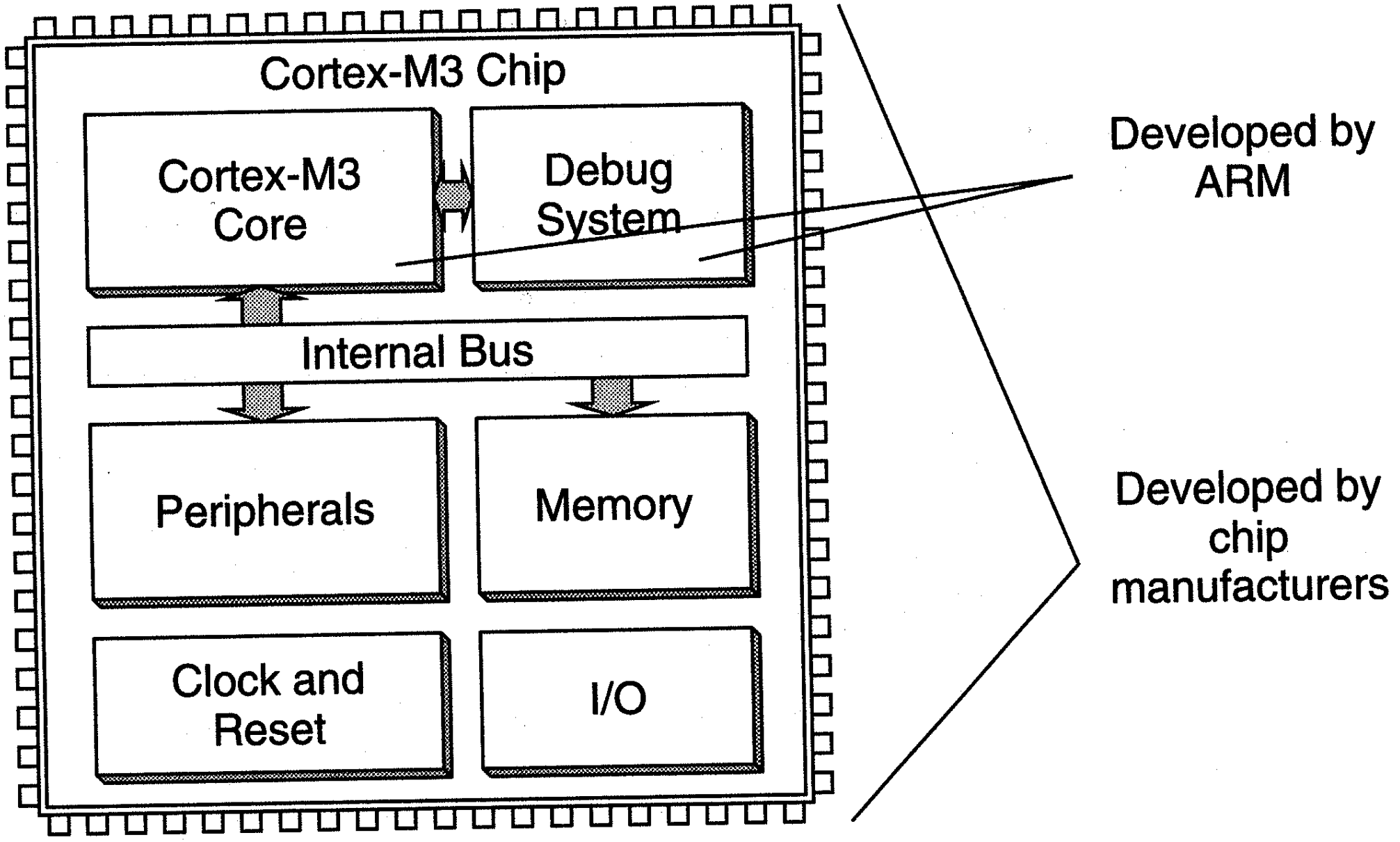
Introduction
organisation - STM32 ? - performance - marché

ARM-Cortex
cpu - adresses - données – interruptions

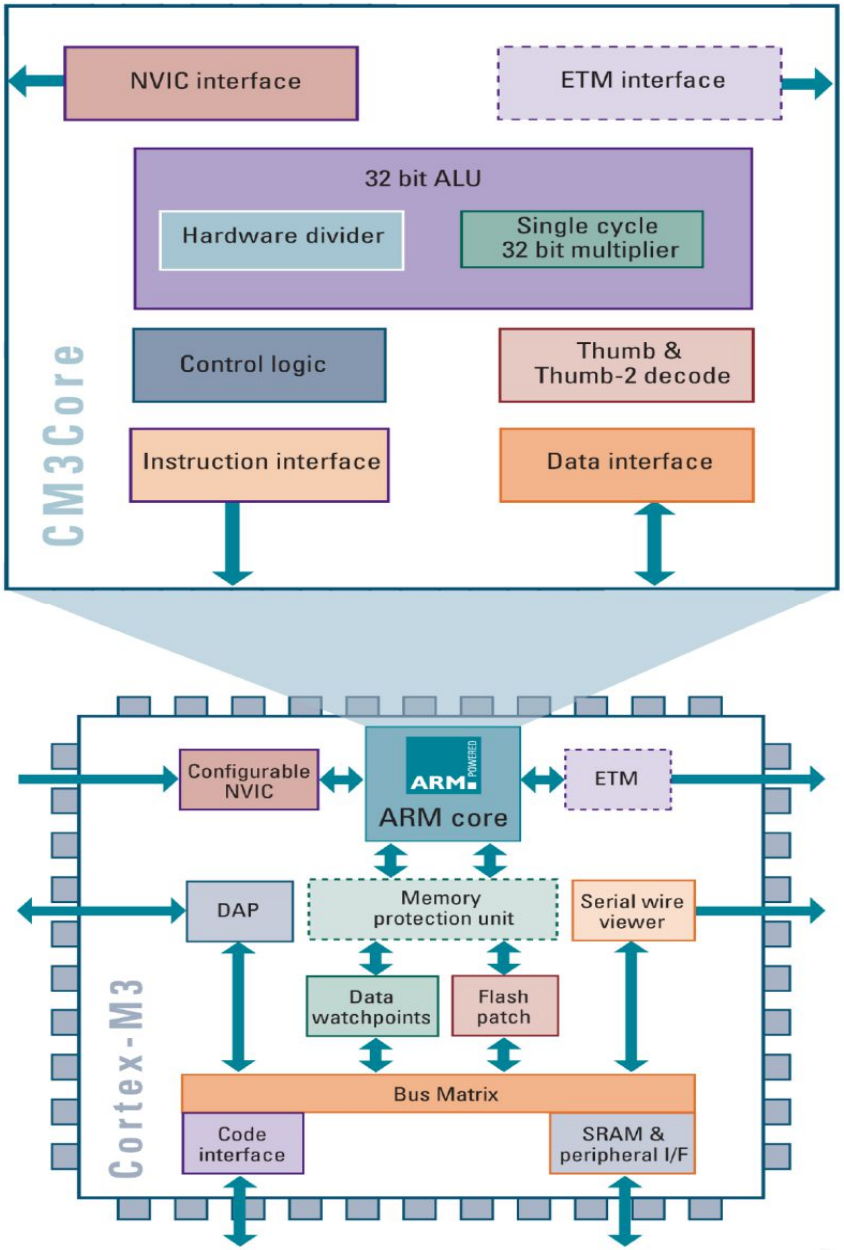
Outil Keil
cible – config – librairie

Périphériques
horloges - gpio - dma - adc – timers

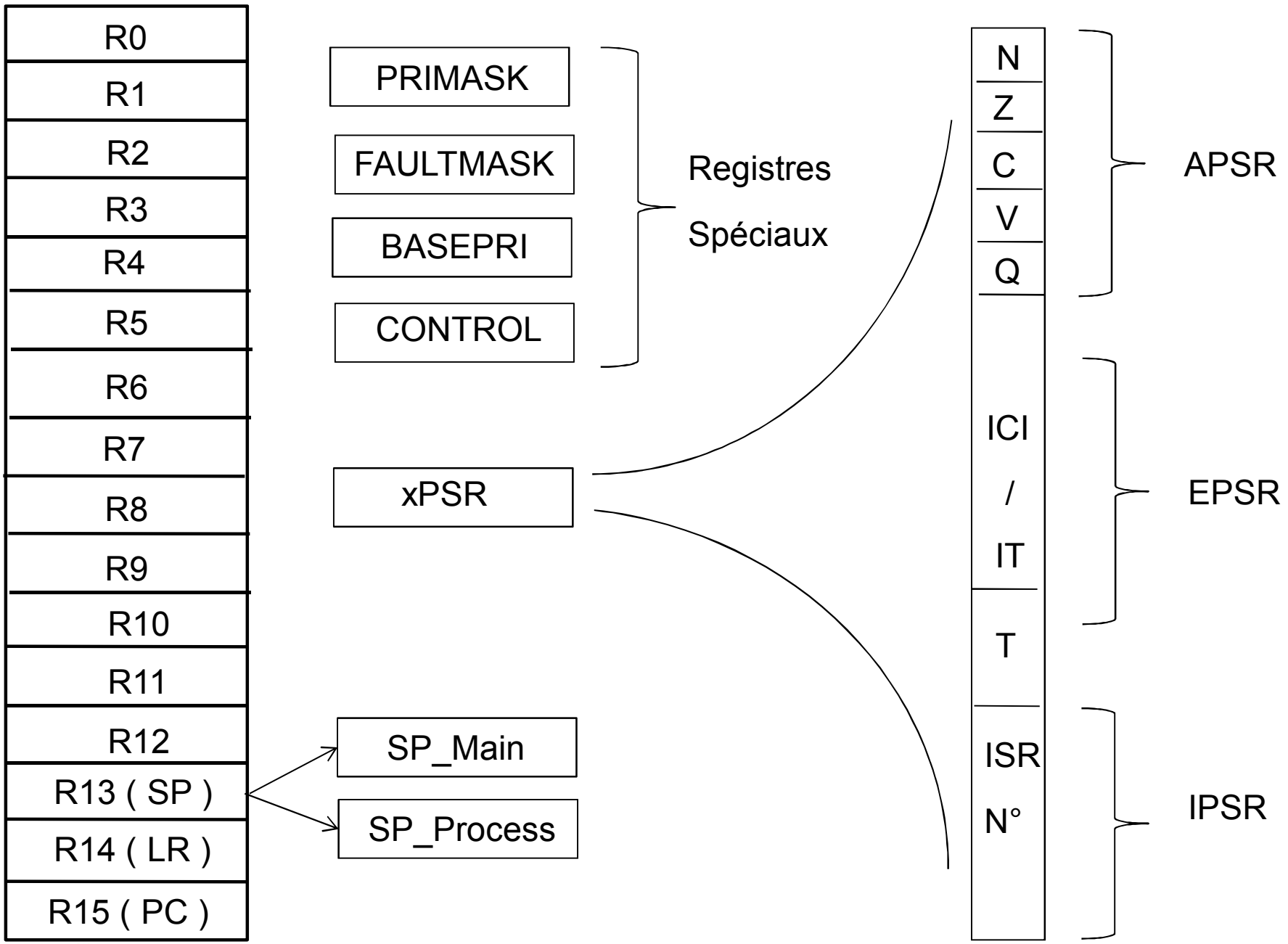
Bus Can
Intérêt - protocole – erreurs – timing – stm32 - application


Cortex-M3 Chip
Cortex-M3 Core
Debug System
Internal Bus
Peripherals
Memory
Clock and Reset
I/O
Developed by ARM
Developed by chip manufacturers

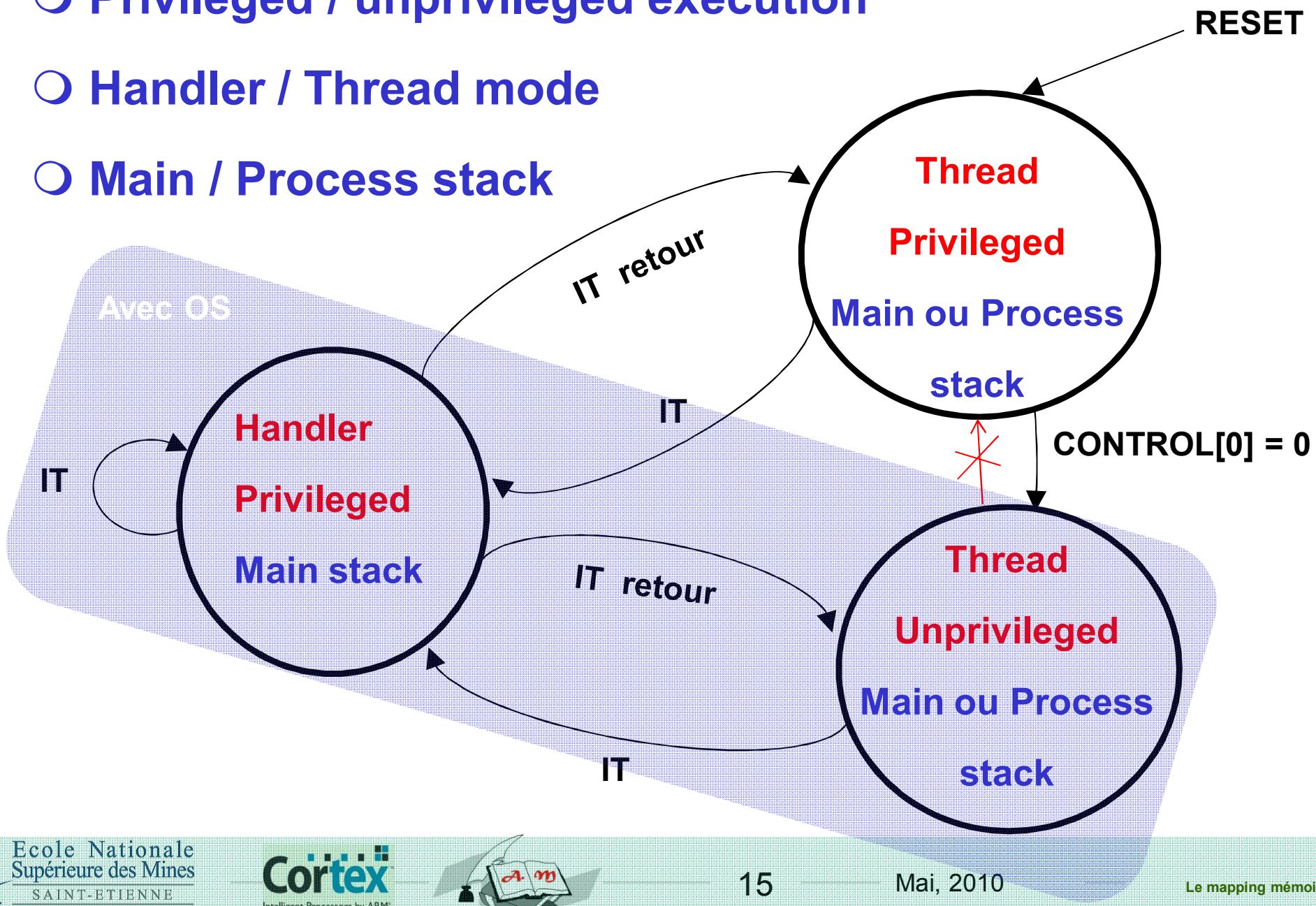
- Harvard architecture
- Pipeline 3 états
- Thumb-2 instructions
- NVIC
- Multiplication 32 b / 1 cycle
- Division 32 b
- Debug systèmes (JTAG)
- Timer système (RTOS)
- Mode basse consommation
- Pas de cache / MMU



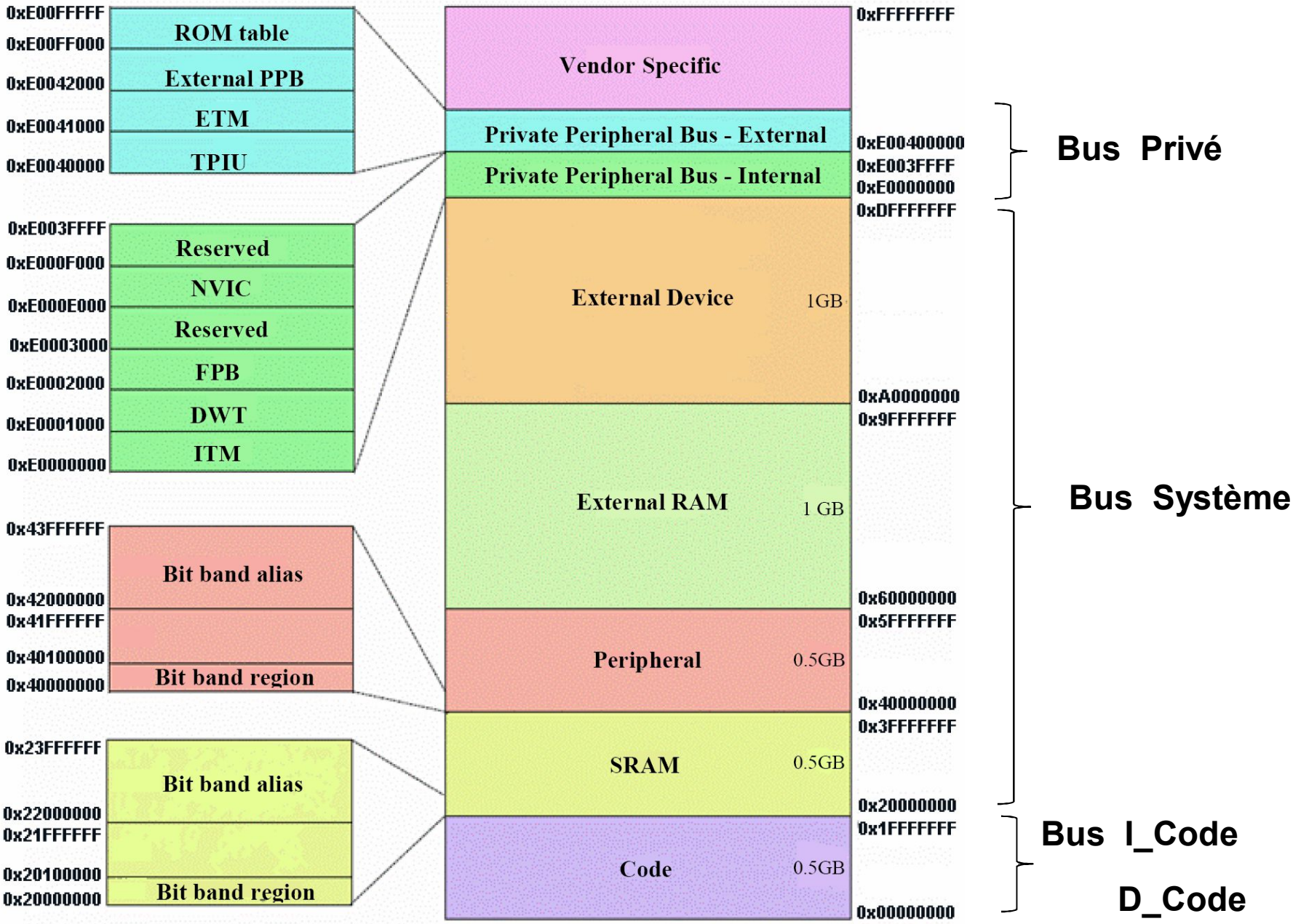
LES REGISTRES



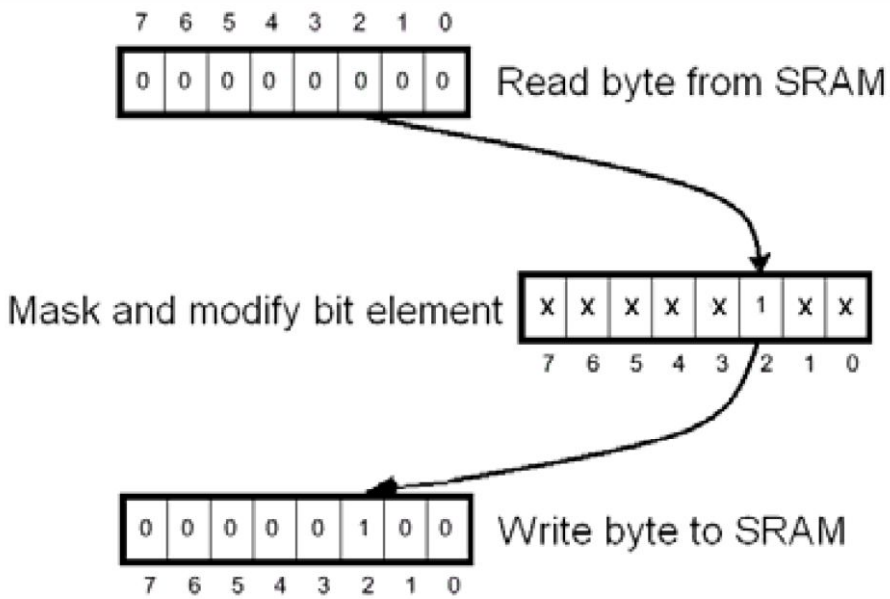
- Privileged / unprivileged exécution
- Handler / Thread mode
- Main / Process stack



MAPPING

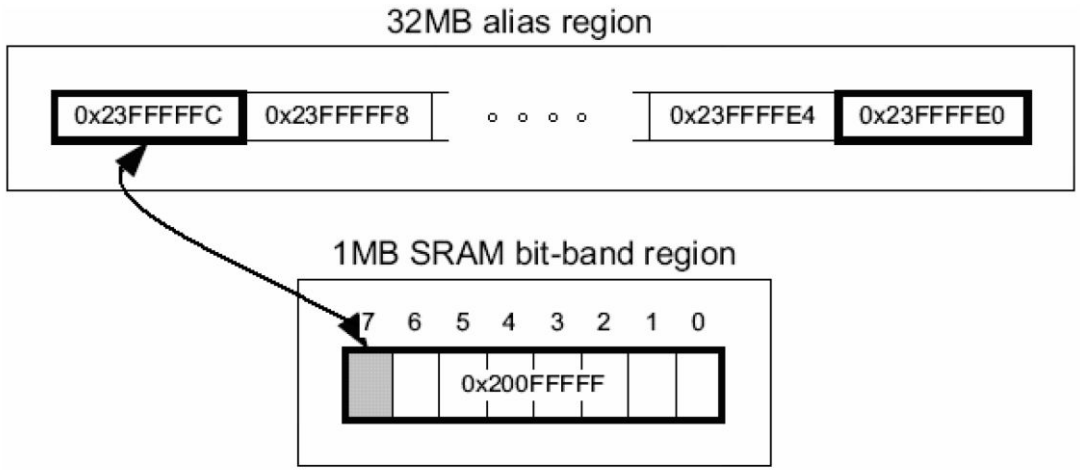

0xE00FFFFF
ROM table
0xE00FF000
External PPB
0xE0042000
ETM
0xE0041000
TPIU
0xE0040000
0xE003FFFF
Reserved
0xE000F000
NVIC
0xE000E000
Reserved
0xE0003000
FPB
0xE0002000
DWT
0xE0001000
ITM
0xE0000000
0x43FFFFFF
Bit band alias
0x42000000
0x41FFFFFF
0x40100000
Bit band region
0x40000000
0x23FFFFFF
Bit band alias
0x22000000
0x21FFFFFF
0x20100000
Bit band region
0x20000000
0xFFFFFFFF
Vendor Specific
Private Peripheral Bus - External
0xE00400000
Private Peripheral Bus - Internal
0xE003FFFF
0xE0000000
0xDFFFFFFF
External Device
1GB
0xA0000000
0x9FFFFFFF
External RAM
1 GB
0x60000000
0x5FFFFFFF
Peripheral
0.5GB
0x40000000
0x3FFFFFFF
SRAM
0.5GB
0x20000000
0x1FFFFFFF
Code
0.5GB
0x00000000
Bus Privé
Bus Système
Bus I_Code
D_Code

BIT ADRESSAGE




```
LDR R0,=0x200FFFFF ; Setup address
MOV R2, #0x4       ; Setup data
LDR R1, [R0]       ; Read
ORR R1, R2         ; Modify bit
STR R1, [R0]       ; Write back result
```

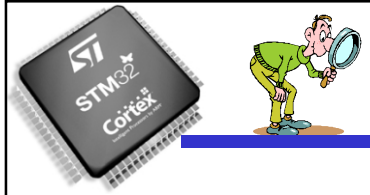
Traditional bit manipulation method




```
LDR R0,=0x23FFFFFC ; Setup address
MOV R1, #0x1       ; Setup data
STR R1, [R0]       ; Write
```

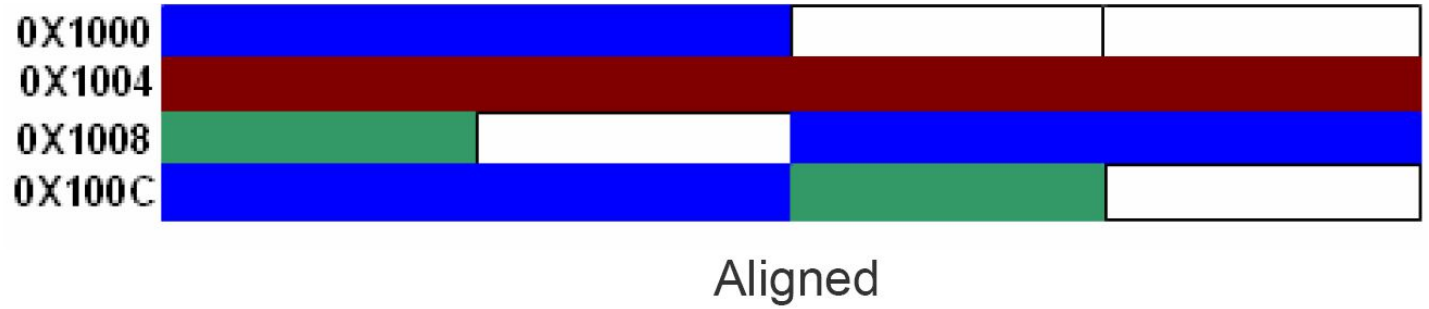
Direct, single cycle access with bit banding

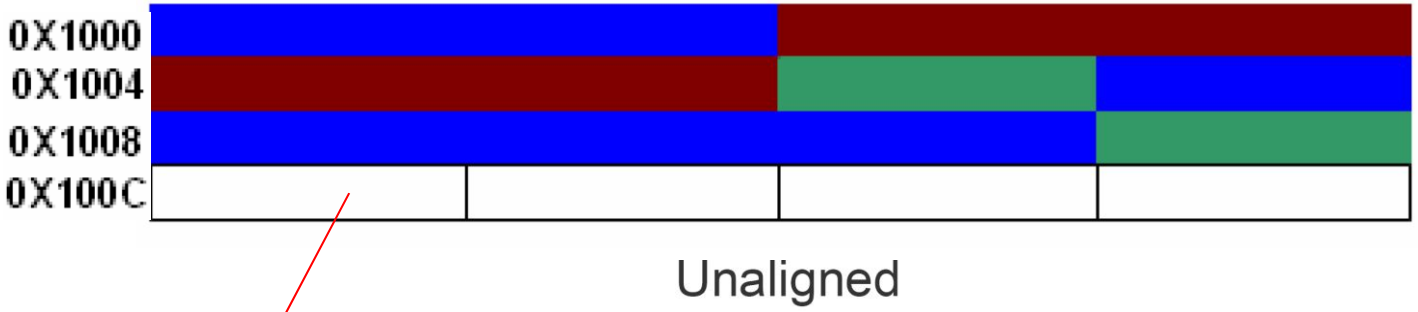
La gestion des bits devient atomique



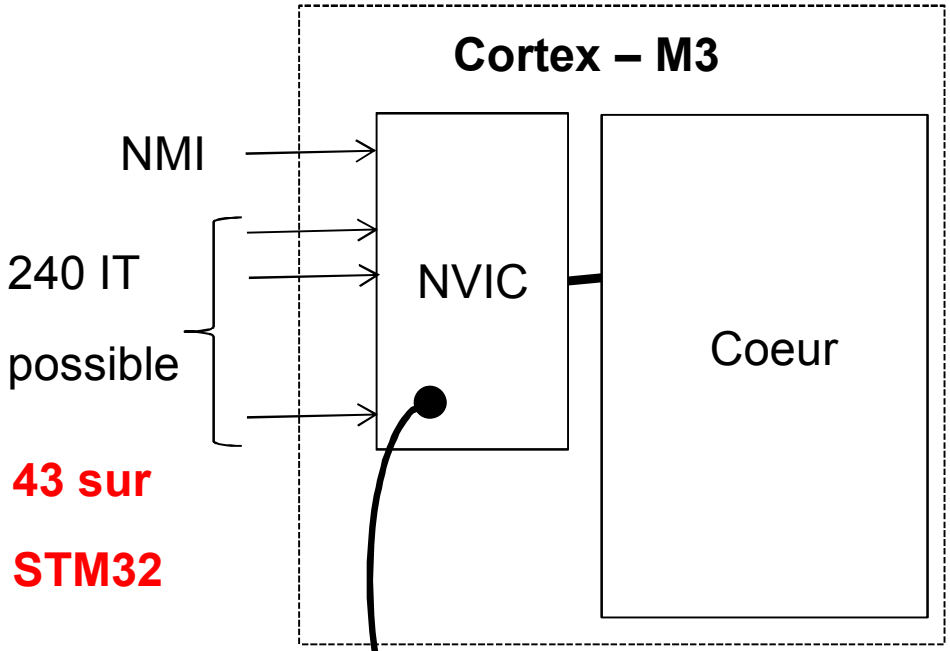

```
 * FONCTION :        Ecrire 0X55555550 à l'adresse 0x20000100
 *                   Forcer le LSB et le bit 2 à 1 en utilisant le bit adressage, ceci de 2 façons
 *                   différentes avec et sans macro.
 *-----------------------------------------------------------------------------------------------------

/*Première solution    */
#define var             (vu32 *) (0x20000100)
#define var_b0          (vu32 *) (    ?????      )
#define var_b2          (vu32 *) (    ?????      )

/*Deuxième solution */
#define RAM_BASE        0x20000000
#define RAM_BB_BASE     0x22000000

/* Private macro ----------------------------------------------------------------------------------------------------*/
#define  Var_ResetBit_BB(VarAddr, Bit)    (*(vu32 *) (RAM_BB_BASE | ((VarAddr - RAM_BASE) << 5) | ((Bit) << 2)) = 0)
#define  Var_SetBit_BB(VarAddr, Bit)      (*(vu32 *) (RAM_BB_BASE | ((VarAddr - RAM_BASE) << 5) | ((Bit) << 2)) = 1)
#define  Var_GetBit_BB(VarAddr, Bit)      (*(vu32 *) (RAM_BB_BASE | ((VarAddr - RAM_BASE) << 5) | ((Bit) << 2)))
```

```
Struct {
        short
        int
        char
        short
        short
        char    };
```





+ GAIN taille 25 % , - Multiple accès bus



- Validation
- Acquittement
- Statu
- Priorité

N°	Type	Priorité
1	Reset	-3 (Haute) fixe
2	NMI	-2 fixe
3	Hard Fault	- 1 fixe
4	MemManage Fault	0
5	Bus Fault	1
6	Usage Fault	2
7-10	Reserved	NA
11	SVcall	3
12	Debug Monitor	4
13	Reserved	NA
14	PendSV	5
15	SYSTICK	6
16	Interupt 0	7
…..	……	…….
256	Interrupt 239	247

- 8 bits de priorité max (4 pour le STM32)
- 2 groupes de priorité : Preempting et sub-priority
- Registre PRIGROUP pour partager les bits de priorités
- Exemple STM32

PRIGROUP (3 Bits)	Binary Point (group.sub)		Preempting Priority (Group Priority)		Sub-Priority	
			Bits	Levels	Bits	Levels
011	4.0	gggg	4	16	0	0
100	3.1	gggs	3	8	1	2
101	2.2	ggss	2	4	2	4
110	1.3	gsss	1	2	3	8
111	0.4	ssss	0	0	4	16

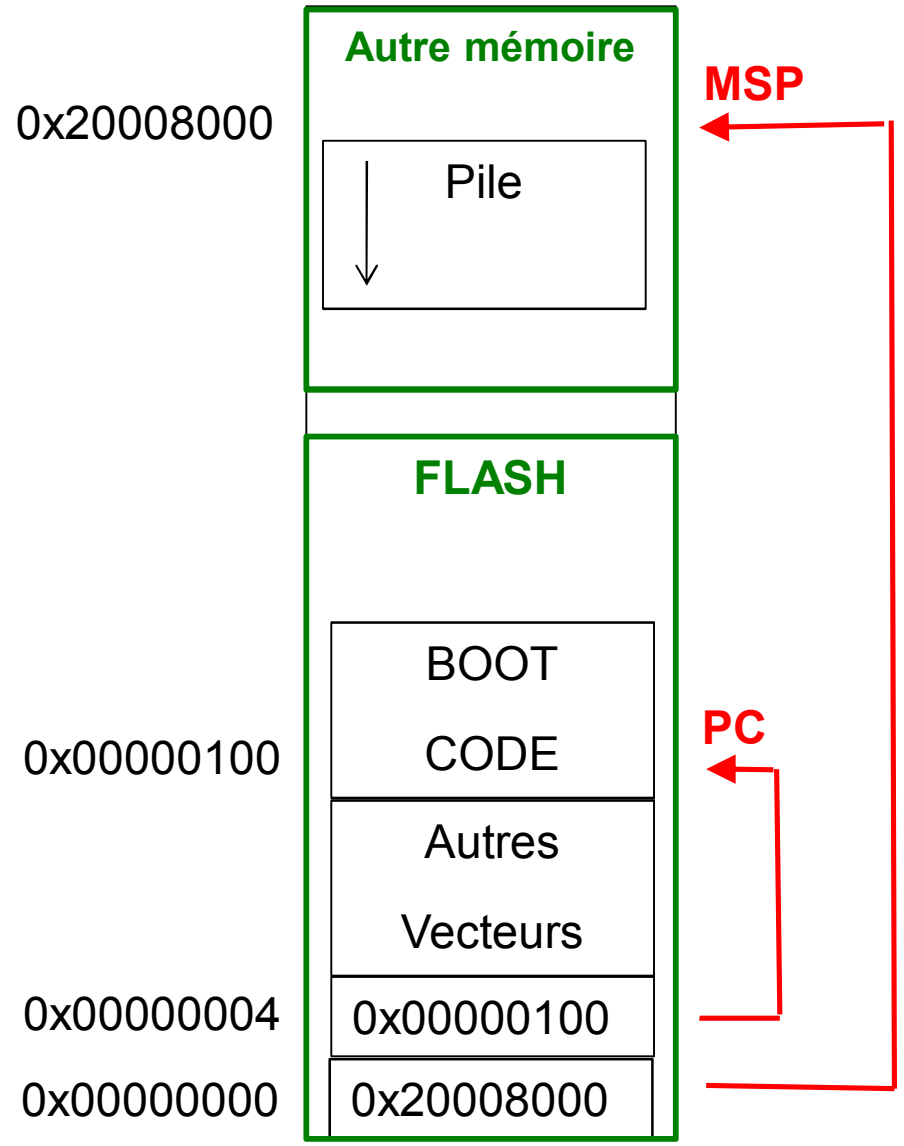
- Masquage des priorités via PRIMASK, FAULMASK, BASEPRI

TABLE DES VECTEURS

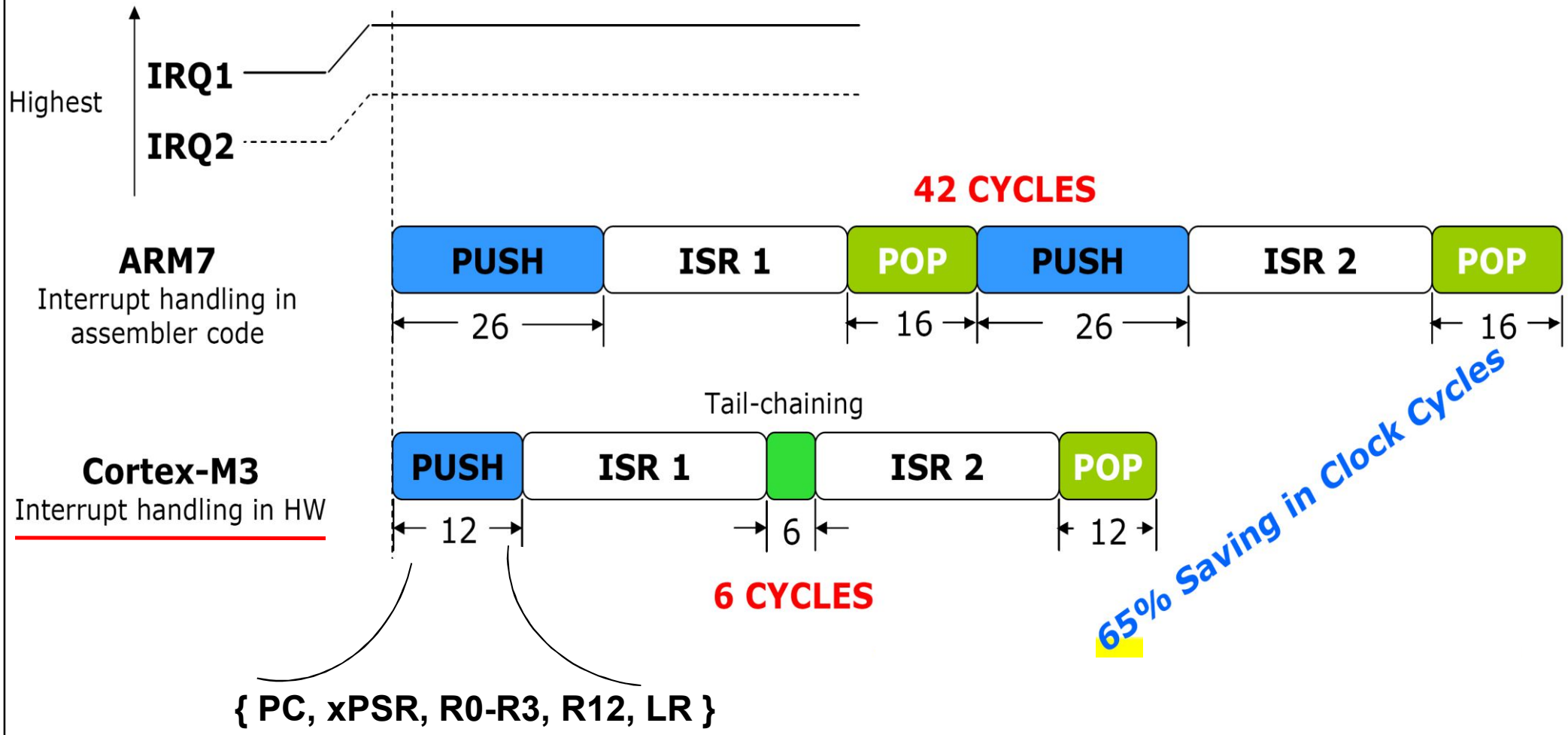
- Relogeable en SRAM
- Contient des adresses

Address	Vector
0x00	Initial Main SP
0x04	Reset
0x08	NMI
0x0C	Hard Fault
0x10	Memory Manage
0x14	Bus Fault
0x18	Usage Fault
0x1C-0x28	Reserved
0x2C	SVCall
0x30	Debug Monitor
0x34	Reserved
0x38	PendSV
0x3C	Systick
40	IRQ0
...	More IRQs

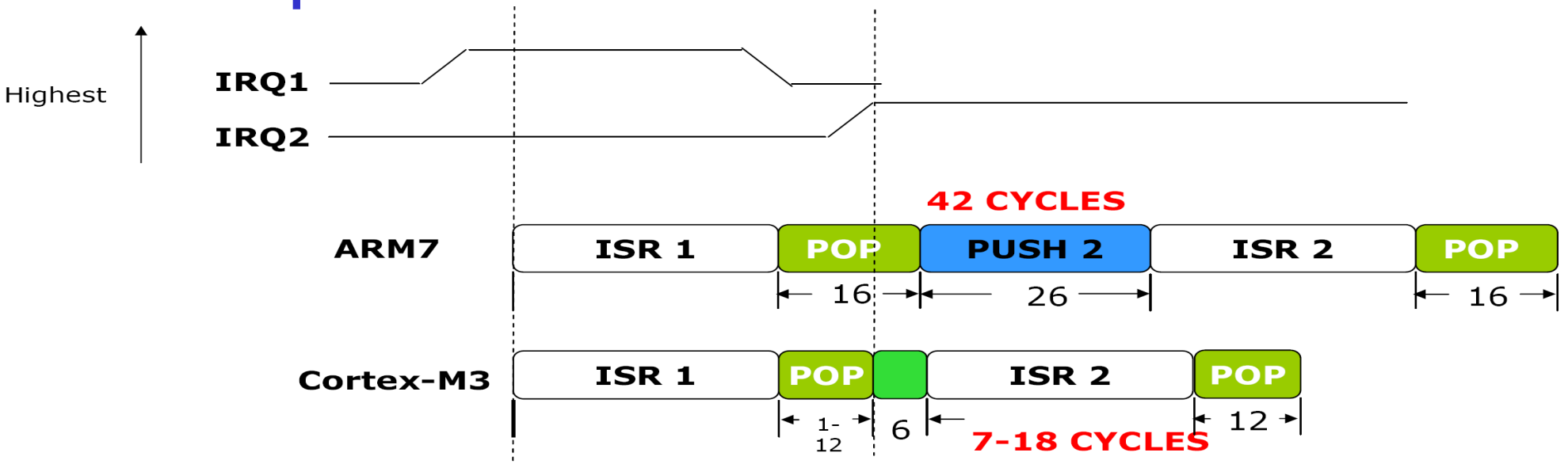
- Séquence RESET



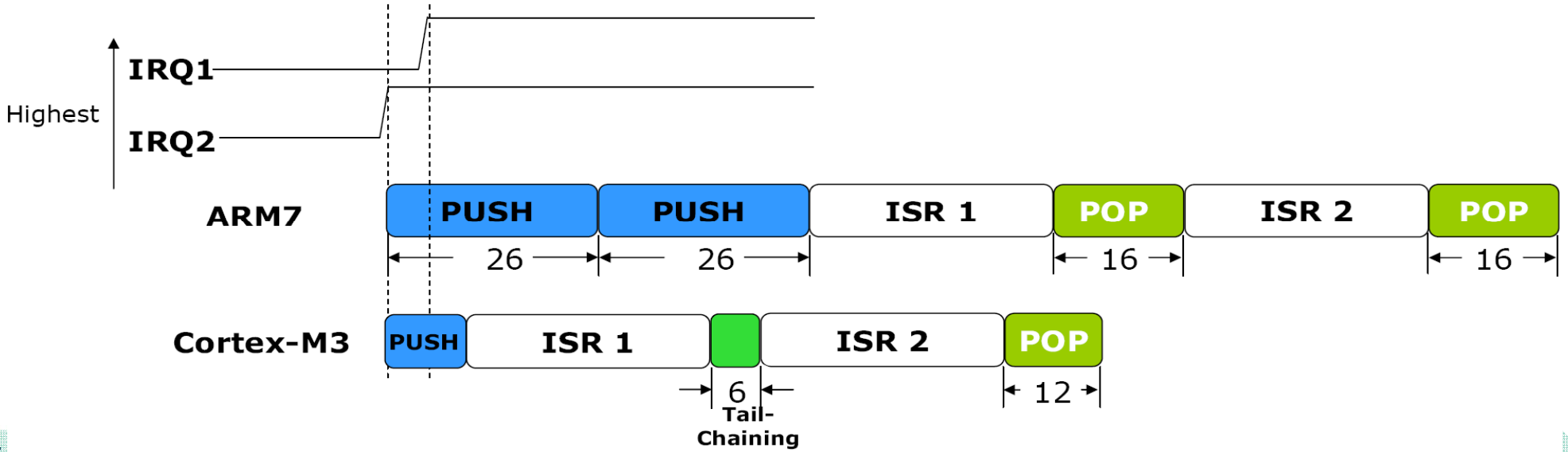
Enchaînement



Pre-emption



Retard



QUIZZ

- Qu'appelle-t-on une IP ?
- Pourquoi dit-on que le Cortex est dédié au temps réel ?
- Qu'appelle-t-on « Bit Band Alias « ? Quel est son intérêt ?
- Je ne peux pas utiliser un masque pour changer un bit sur le Cortex, vrai ou faux ?
- Pourquoi les données unaligned ralentisse les accès bus ?
- Comment est initialisé le Main Stack Pointer au démarrage ?
- Où sont géré les priorités des interruptions ?
- Combien d'interruptions peut gérer le NVIC? Combien sont utilisées sur le STM 32 ?

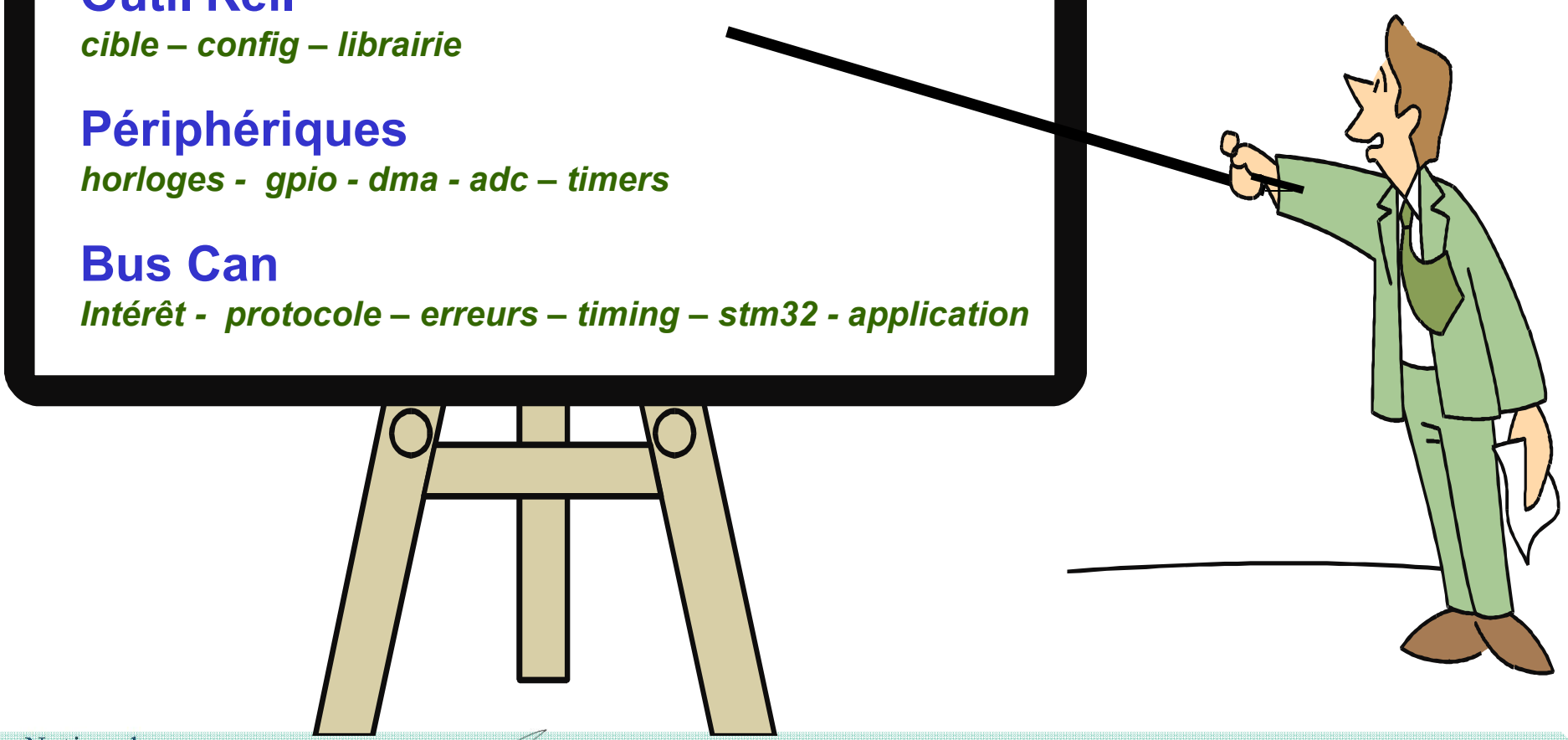
Introduction
organisation - STM32 ? - performance - marché

ARM-Cortex
cpu - adresses - données – interruptions

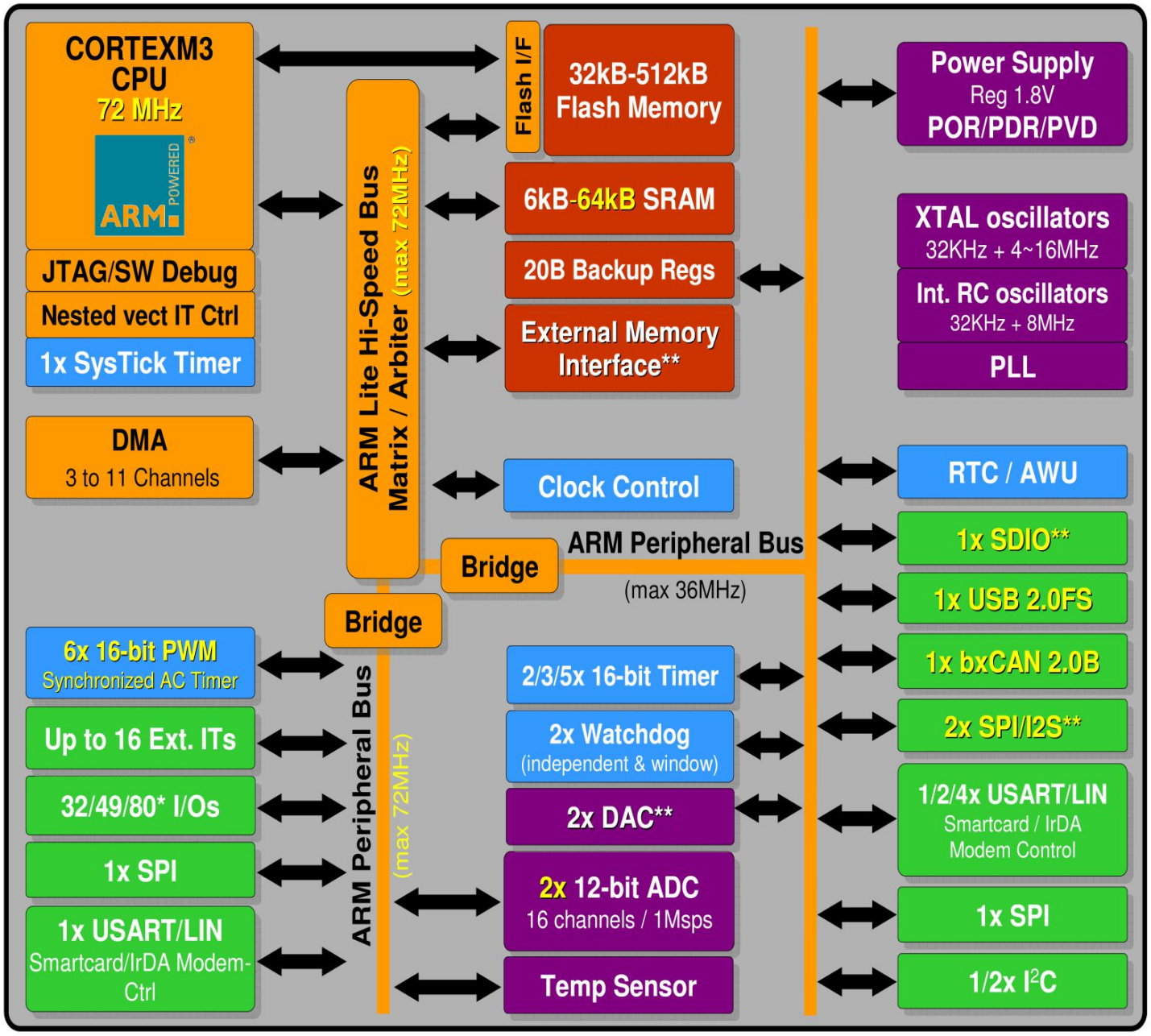
Outil Keil
cible – config – librairie

Périphériques
horloges - gpio - dma - adc – timers

Bus Can
Intérêt - protocole – erreurs – timing – stm32 - application

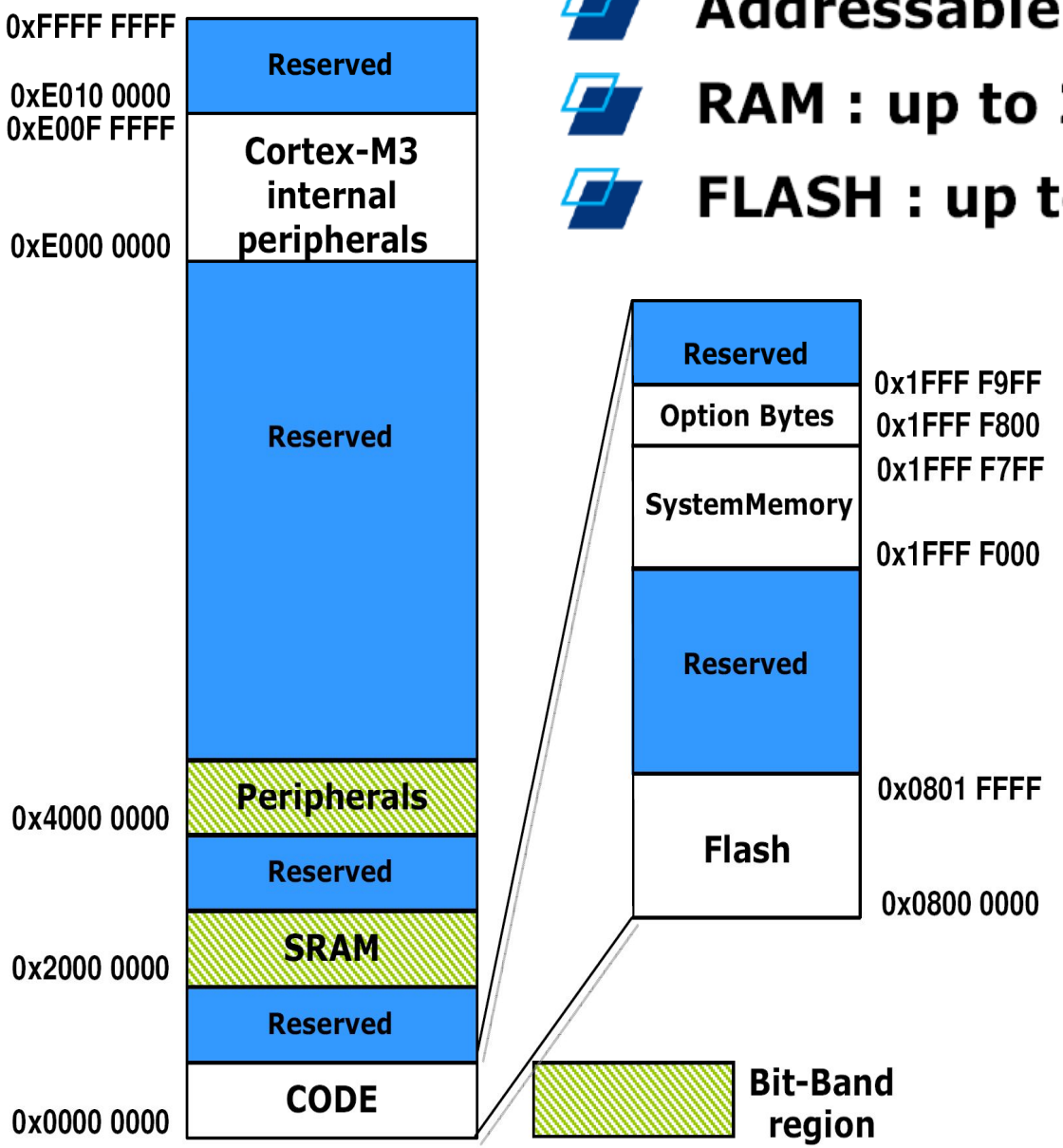


- 2V-3.6V Supply
- Cortex-M3 Core 72MHz 90DMIPs
- 5V tolerant I/Os
- Excellent safe clock modes
- Low-power modes with wake-up
- Internal RC
- Embedded reset
- -40/+105°C

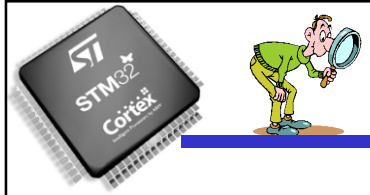


MAPPING Cortex M3

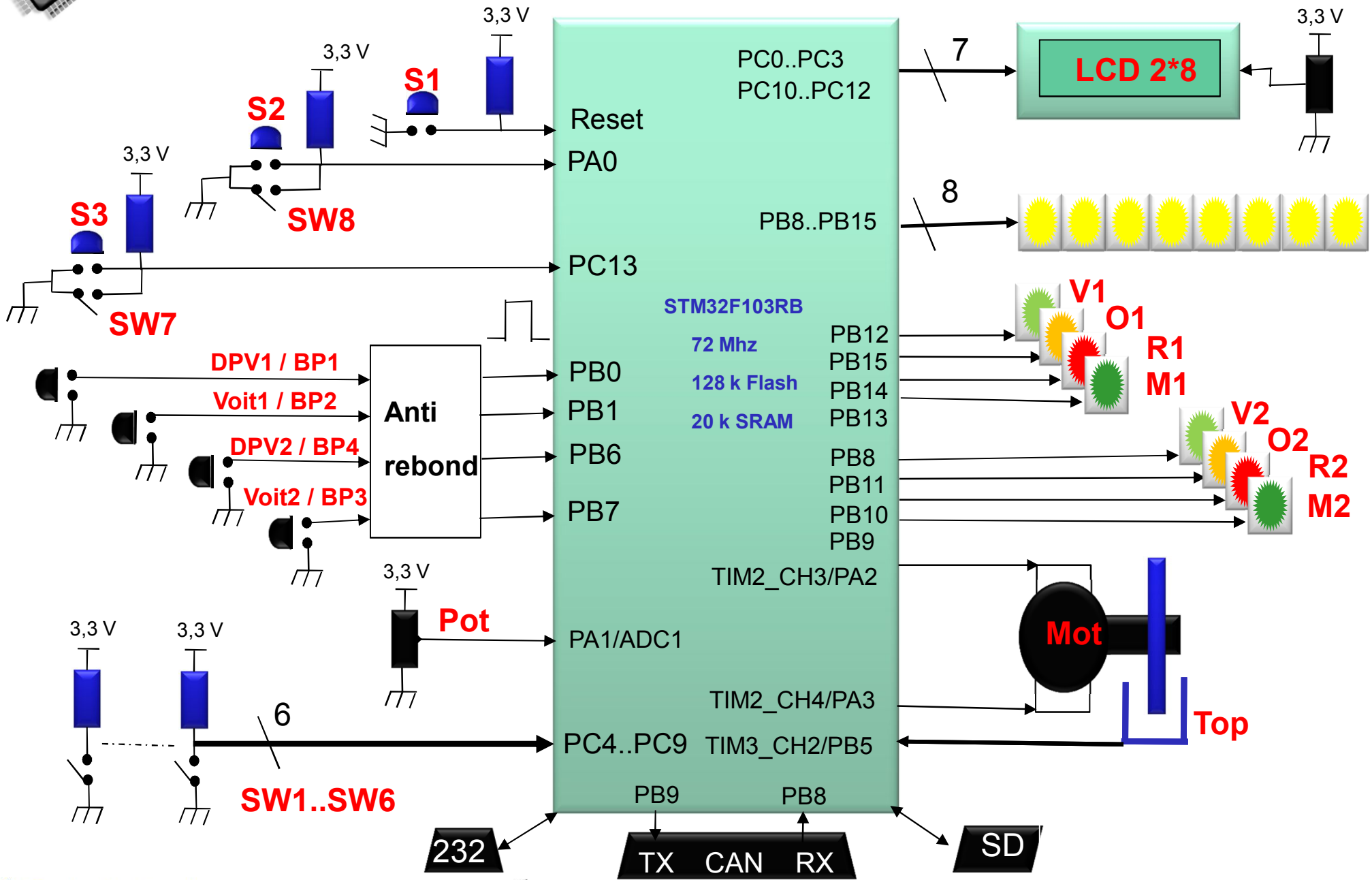
- Addressable memory space of 4 GBytes
- RAM : up to 20 kBytes
- FLASH : up to 128 kBytes

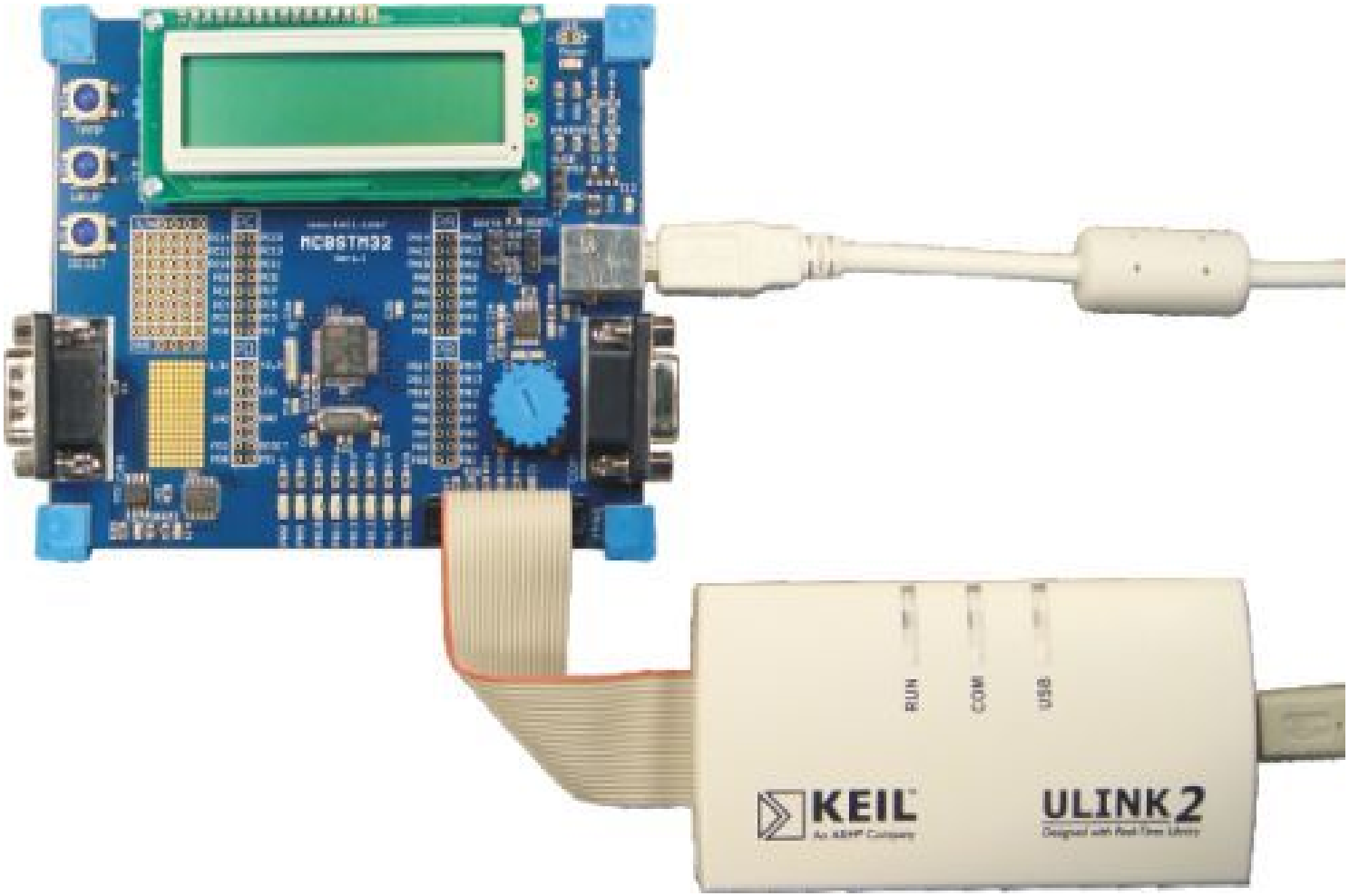


BOOT Mode Selection Pins		Boot Mode	Aliasing
BOOT1	BOOT0		
x	0	User Flash	User Flash is selected as boot space
0	1	SystemMemory	SystemMemory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space



OUTILS : cible – config – librairie


3,3 V
S1
S2
S3
SW8
SW7
Reset
PA0
PC13
PC0..PC3
PC10..PC12
7
LCD 2*8
PB8..PB15
8
STM32F103RB
72 Mhz
128 k Flash
20 k SRAM
V1
O1
R1
M1
V2
O2
R2
M2
DPV1 / BP1
Voit1 / BP2
DPV2 / BP4
Voit2 / BP3
Anti
rebond
PB0
PB1
PB6
PB7
PB12
PB15
PB14
PB13
PB8
PB11
PB10
PB9
TIM2_CH3/PA2
Pot
PA1/ADC1
Mot
Top
TIM2_CH4/PA3
TIM3_CH2/PB5
6
PC4..PC9
SW1..SW6
PB9
PB8
232
TX
CAN
RX
SD



Alimentation via USB

Téléchargement via USB

OUTILS : cible – config – librairie

Options for Target 'MCBSTM32'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

STMicroelectronics STM32F103RB

Xtal (MHz): 8.0

Operating system:

Code Generation

- [ ] Use Cross-Module Optimization
- [x] Use MicroLIB

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
[ ]	ROM1:			( )
[ ]	ROM2:			( )
[ ]	ROM3:			( )
	on-chip			
[x]	IROM1:	0x8000000	0x20000	(•)
[ ]	IROM2:			( )

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
[ ]	RAM1:			[ ]
[ ]	RAM2:			[ ]
[ ]	RAM3:			[ ]
	on-chip			
[x]	IRAM1:	0x20000000	0x5000	[ ]
[ ]	IRAM2:			[ ]

OK | Annuler | Defaults | Aide

OUTILS : cible – config – librairie

Options for Target 'MCBSTM32'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

Use Simulator — Limit Speed to Real-Time — Settings — Use: ULINK Cortex Debugger — Settings

Load Application at Startup	Load Application at Startup
Run to main()	Run to main()
Initialization File:	Initialization File:
	.\STM32DBG.ini
Restore Debug Session Settings	Restore Debug Session Settings
Breakpoints, Toolbox	Breakpoints, Toolbox
Watchpoints & PA	Watchpoints
Memory Display	Memory Display
CPU DLL: SARMCM3.DLL	Driver DLL: SARMCM3.DLL
Parameter:	Parameter:
Dialog DLL: DARMSTM.DLL	Dialog DLL: TARMSTM.DLL
Parameter: -pSTM32F103RB	Parameter: -pSTM32F103RB

OK — Annuler — Defaults — Aide

Cortex-M Target Driver Setup

Debug | Trace | Flash Download

ULINK USB - JTAG/SW Adapter

Serial No: V0282PAE

ULINK Version: ULINK2

Device Family: Cortex-M

Firmware Version: V1.36

SWJ Port: SW

Max Clock: 5MHz

SW Device

	IDCODE	Device Name
SWDIO	0x1BA01477	ARM CoreSight SW-DP

Move — Up — Down

Automatic Detection — ID CODE:

Manual Configuration — Device Name:

Add — Delete — Update — IR len:

Debug

Connect & Reset Options

Connect: with Pre-reset — Reset: Autodetect

Reset after Connect

Cache Options

Cache Code

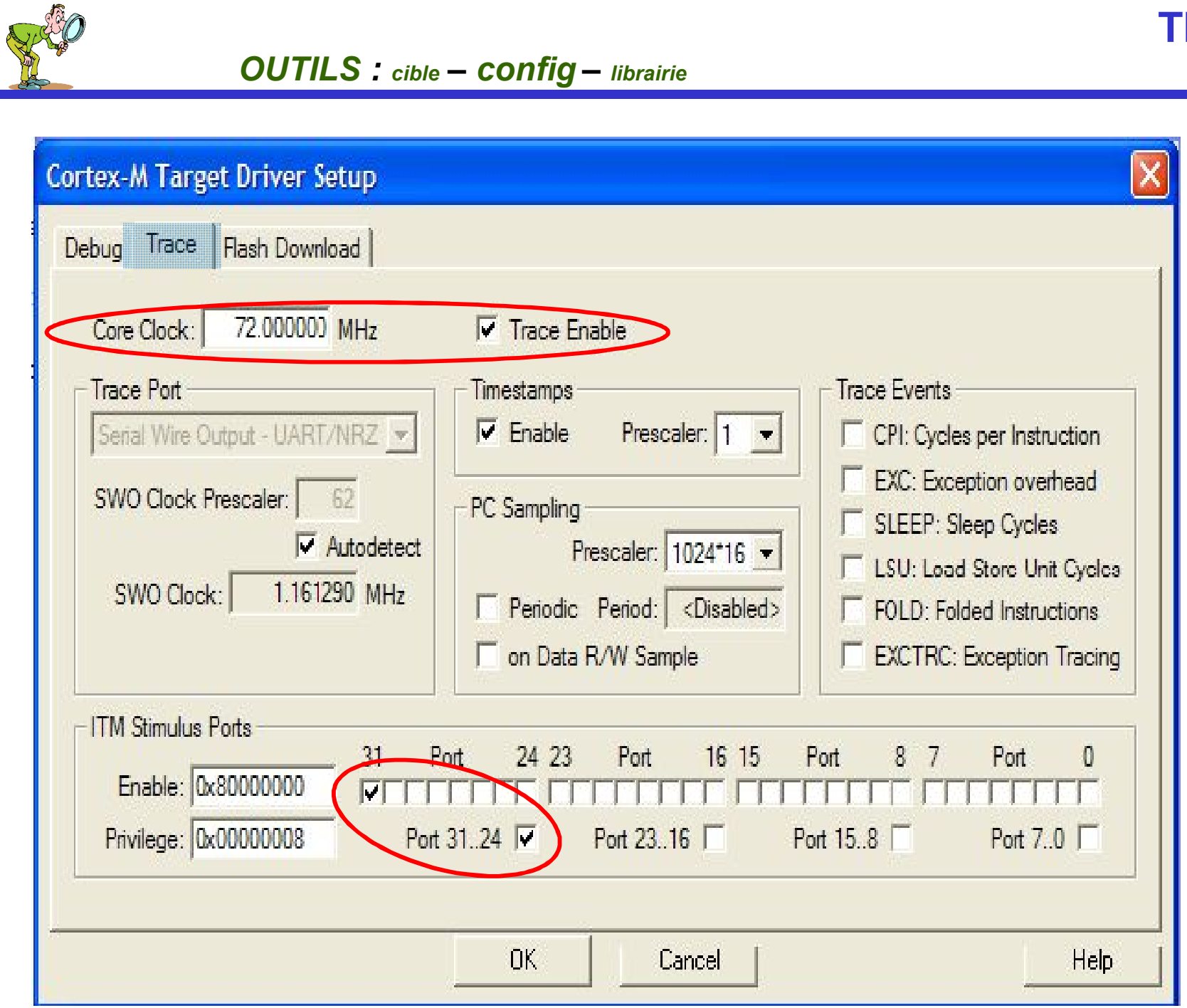
Cache Memory

Download Options

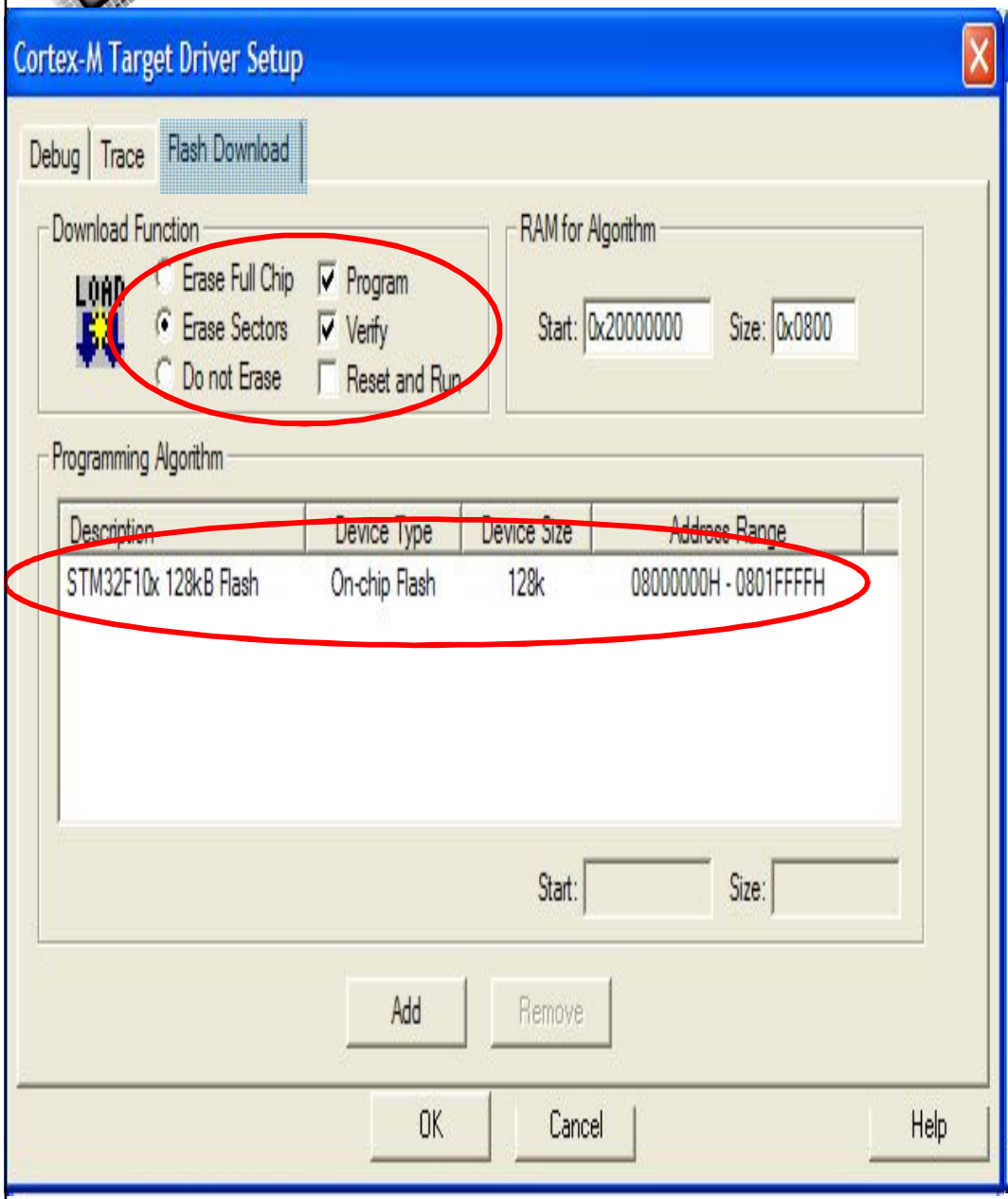
Verify Code Download

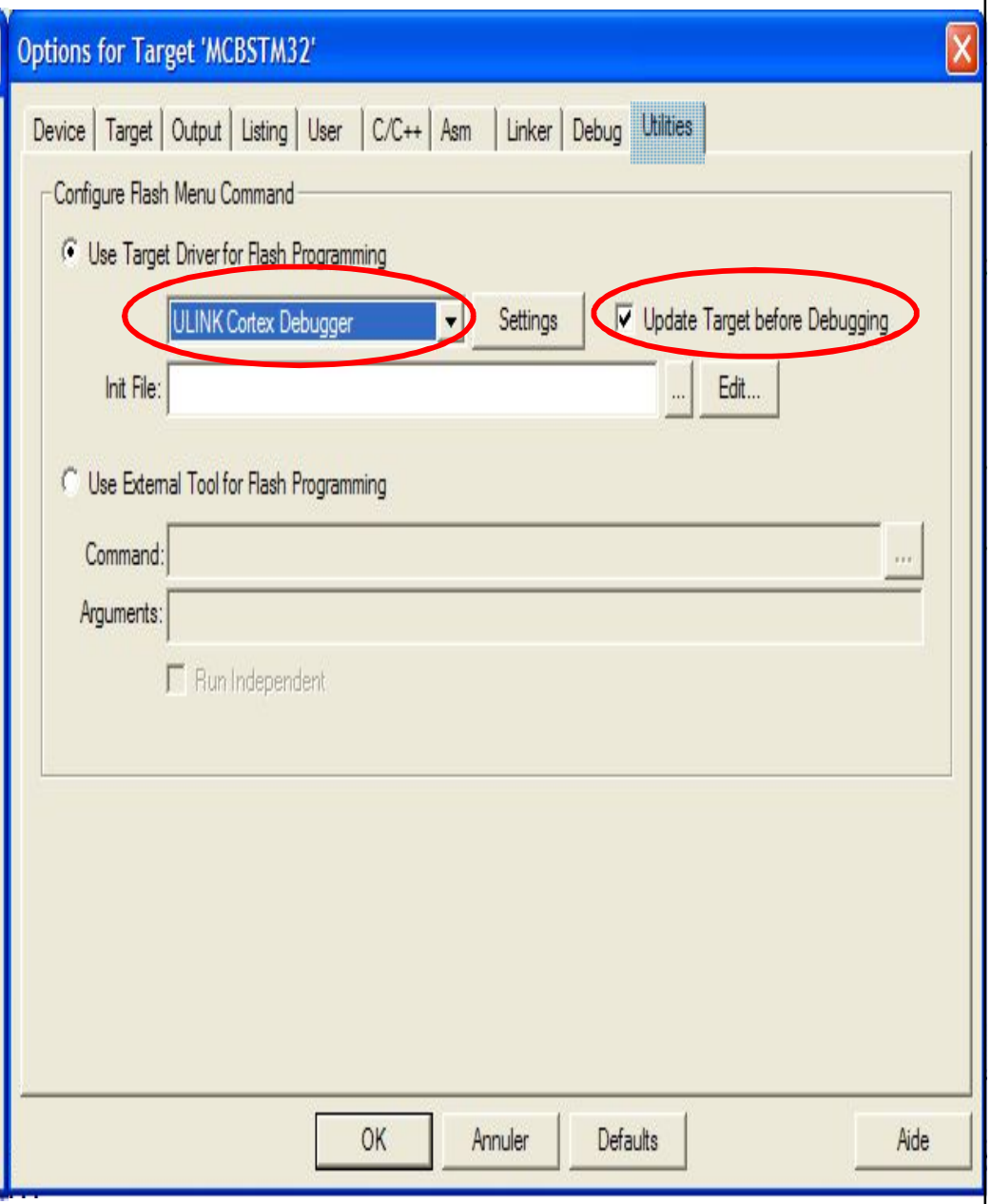
Download to Flash

OK — Cancel — Help

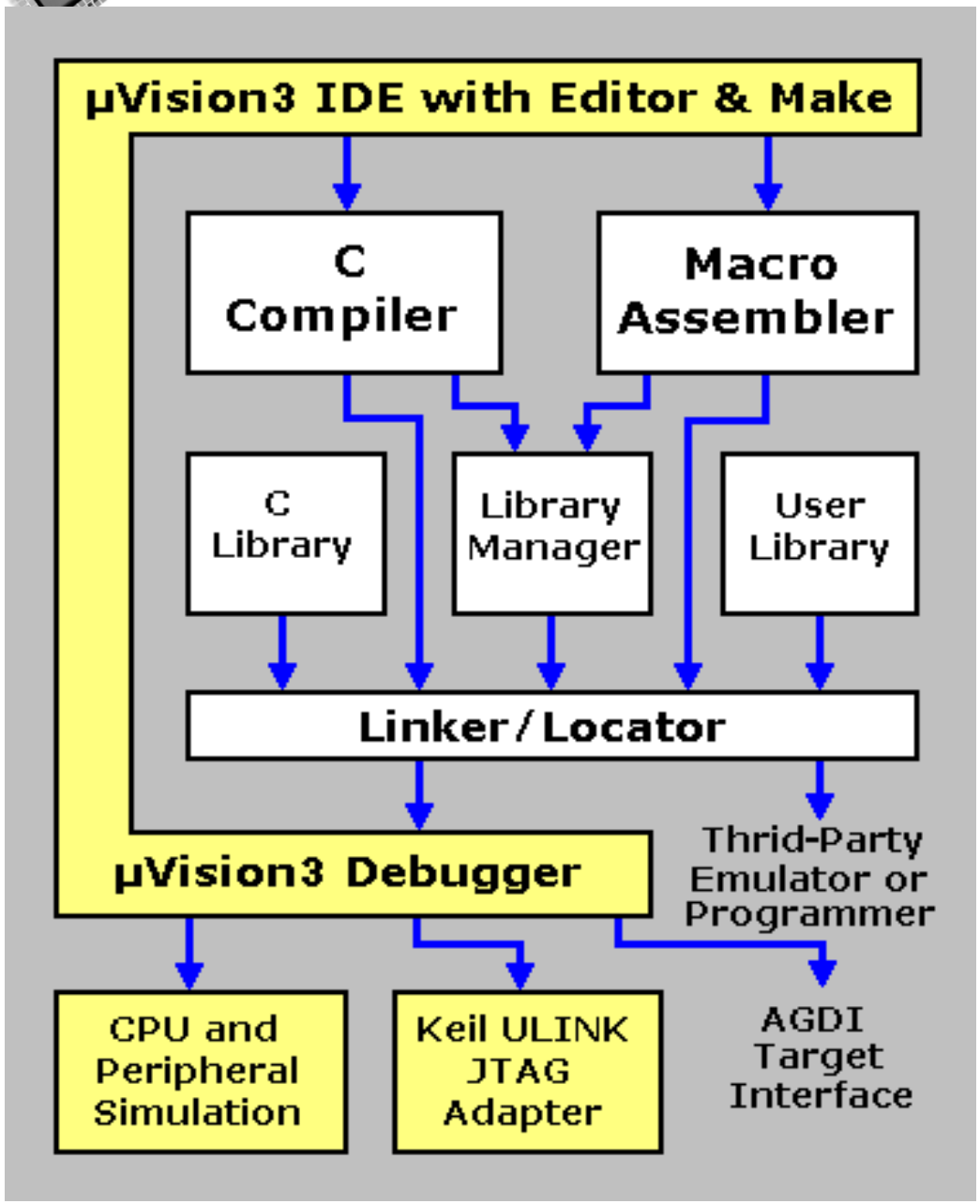

Cortex-M Target Driver Setup
Debug
Trace
Flash Download
Core Clock: 72.000000 MHz
Trace Enable
Trace Port
Serial Wire Output - UART/NRZ
SWO Clock Prescaler: 62
Autodetect
SWO Clock: 1.161290 MHz
Timestamps
Enable
Prescaler: 1
PC Sampling
Prescaler: 1024*16
Periodic
Period: <Disabled>
on Data R/W Sample
Trace Events
CPI: Cycles per Instruction
EXC: Exception overhead
SLEEP: Sleep Cycles
LSU: Load Store Unit Cycles
FOLD: Folded Instructions
EXCTRC: Exception Tracing
ITM Stimulus Ports
31 Port 24 23 Port 16 15 Port 8 7 Port 0
Enable: 0x80000000
Privilege: 0x00000008
Port 31..24
Port 23..16
Port 15..8
Port 7..0
OK
Cancel
Help

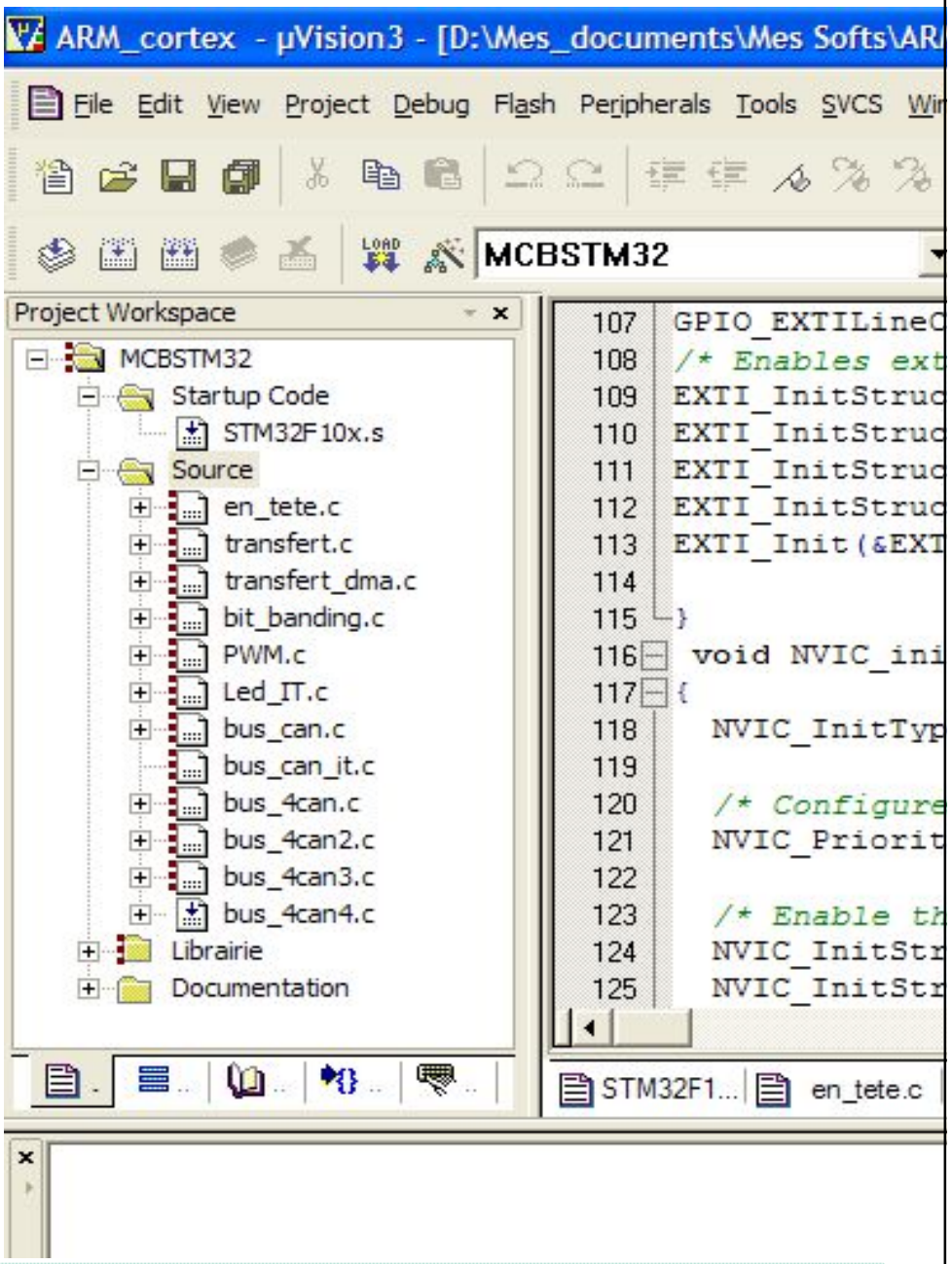
OUTILS : cible – config – librairie





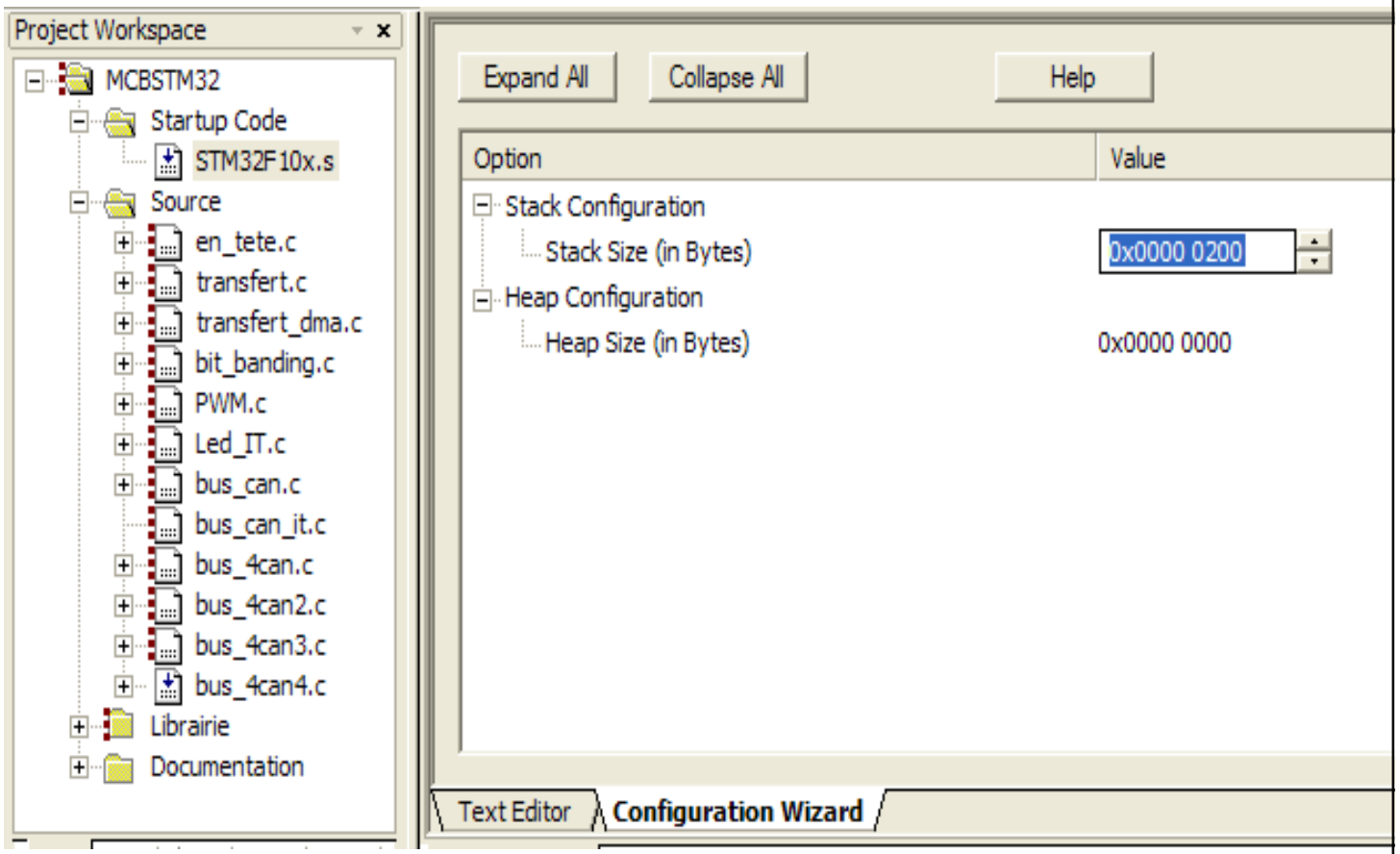
OUTILS : cible – config – librairie





OUTILS : cible – config – librairie

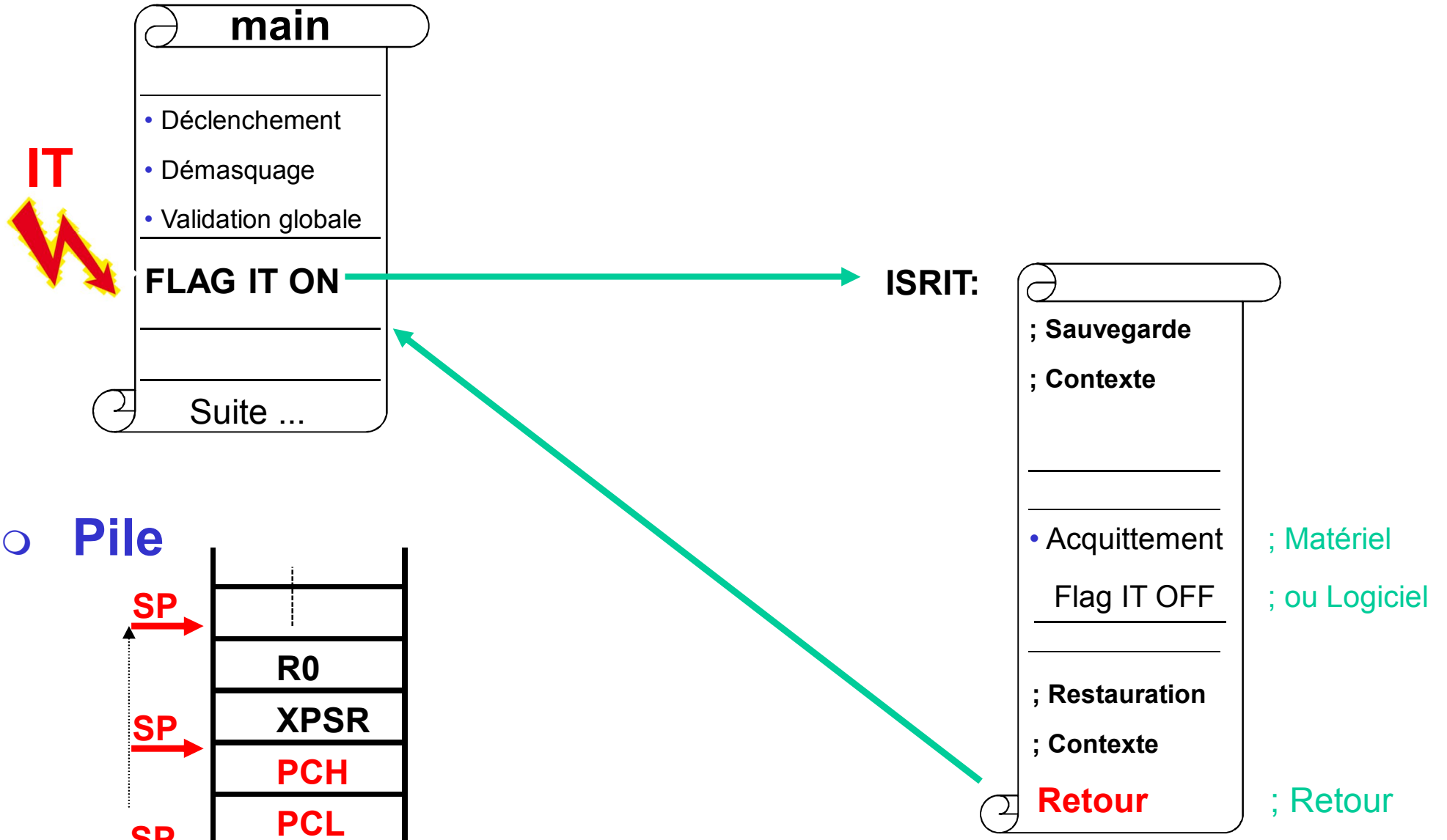
- Init Stack
- Init Heap
- Init Vecteurs d'interruption




```
Reset_Handler
        EXPORT  Reset_Handler    [WEAK]
        B    main
```

```
Autres_Handler
        EXPORT  Autres_Handler    [WEAK]
        B    .
```

Mécanisme

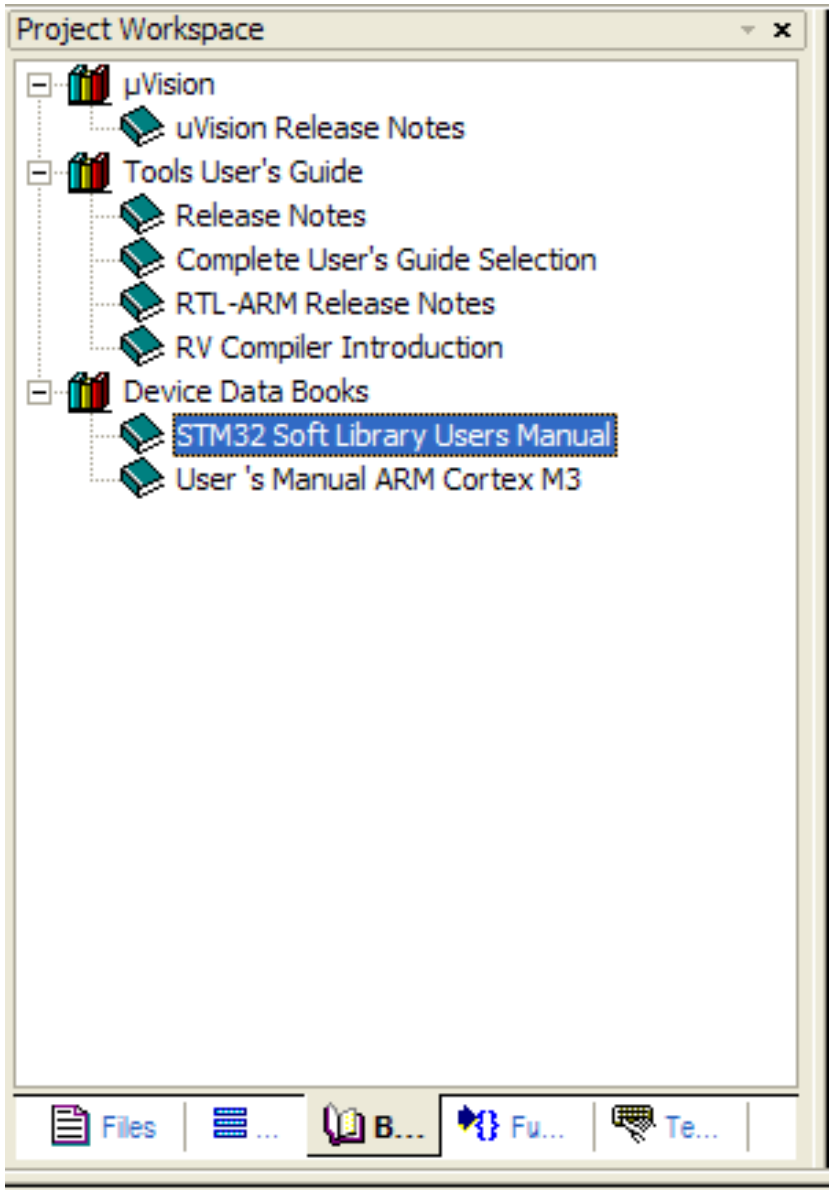


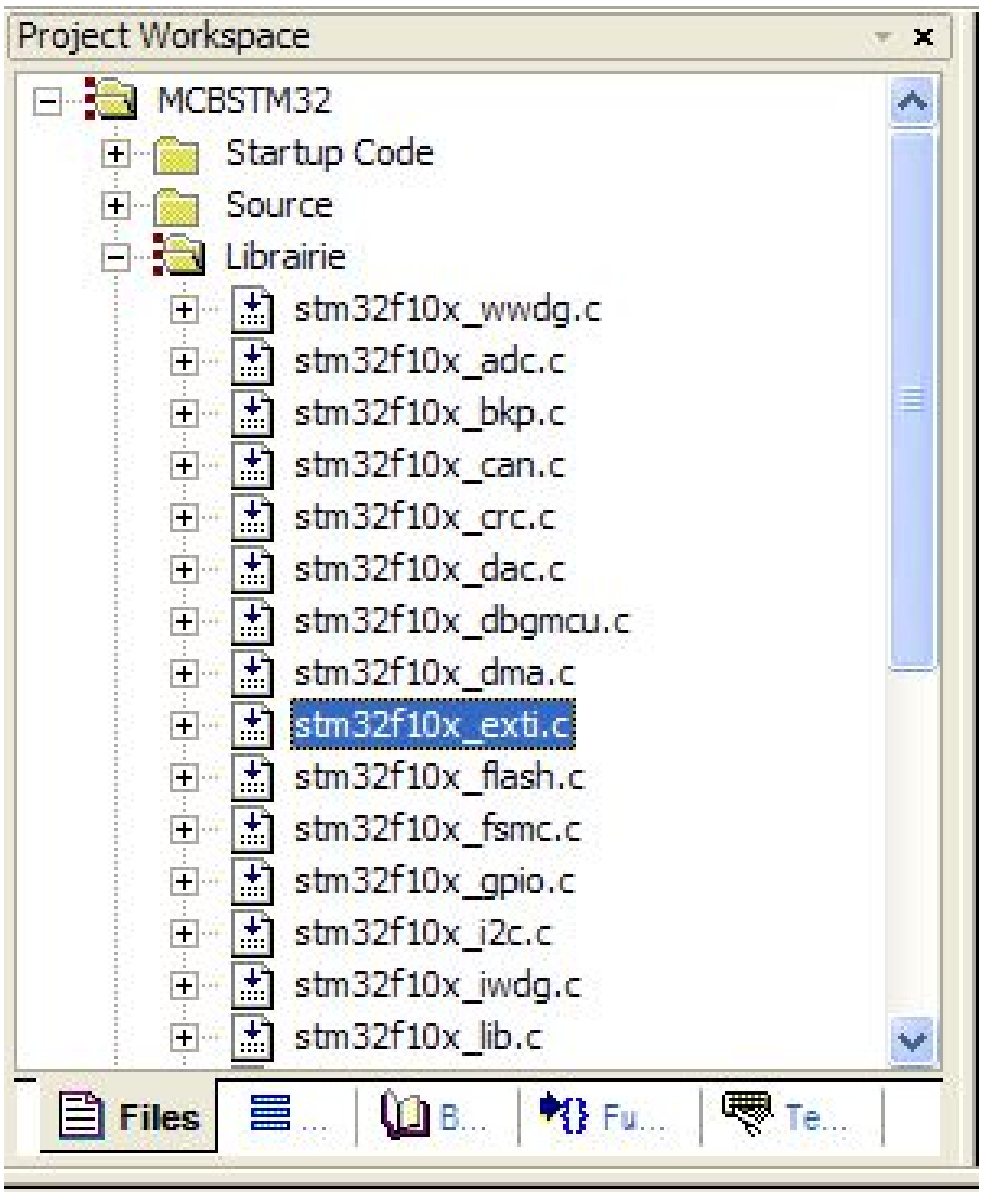
Pile

Exemple

```
/*****************************************************************/
/*          Exemple d'interruption INT1                          */
/*****************************************************************/

void  EXTI1_IRQHandler(void)        {
          cpt_top++ ;
          EXTI_ClearITPendingBit(EXTI_Line1); // Acquitement
                                              // EXTI->PR |= (1<<1);
}
```

OUTILS : cible – config – librairie





Stm32f10x_map.h

```
typedef struct
{
  vu32 IMR;
  vu32 EMR;
  vu32 RTSR;
  vu32 FTSR;
  vu32 SWIER;
  vu32 PR;
} EXTI_TypeDef;

#define PERIPH_BASE                ((u32)0x40000000)
#define APB2PERIPH_BASE    (PERIPH_BASE + 0x10000)
#define EXTI_BASE          (APB2PERIPH_BASE + 0x0400)
#ifdef  _EXTI
#define EXTI               ((EXTI_TypeDef *) EXTI_BASE)
#endif /* _EXTI */
```

Stm32f10x_conf.h

```
# define  _EXTI
```

EXTI →

0X40010400

Interrupt Mask Register
Event Mask Register
Rising Trigger Selection Register
Falling Trigger Selection Register
SoftWare Interrupt Event Register
Pending Register

Utilisation

- Les registres des interfaces de périphériques

- Les variables globales modifiées par une routine d'interruption.

Exemple

```
                #define     RX_CHAR                 ( * ( volatile unsigned char * )  ( 0x8000  ) )
volatile        Int         End_Message = FALSE;


                void main ( void )
                {
                ....
                while  (! End_Message) ;
                 ....

                void Isr_Receive ( void )                                   // interruption
                {
                .....
                if ( RX_CHAR == ETX ) { End_Message = TRUE;}
                .....
                }
```

OUTILS : cible – config – librairie

```
void EXTI_init () {
    EXTI_InitTypeDef   EXTI_InitStructure;
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource1); /* Selects PA.1 as EXTI */
    EXTI_InitStructure.EXTI_Line = EXTI_Line1;                  /*  Line 1 */
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;   // Enables external lines 0 interrupt
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;  // generation on falling edge
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);                                 }

void NVIC_init(void)   {
    NVIC_InitTypeDef   NVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); /* Configure 2 bits for preemption priority */
    NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQChannel; /* Enable the EXTI1 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);                                 }
```

QUIZZ

- Quelle est la capacité d'adressage du cortex M3 ?
- Donner la taille de la RAM et de la Flash du STM32 ?
- La Flash est plus lente que le cortex pourtant les 2 communiquent, pourquoi ?
- Le STM32 utilise la liaison Jtag ou la liaison serial wire pour le debug ?
- Rôle d'une trace ? Intérêt ?
- Citer 2 manières de programmer les périphériques du cortex ?
- Rôle du fichier STM32F10x.s ?
- Sur le STM32 que se passe-t-il si une interruption non initialisé survient ?

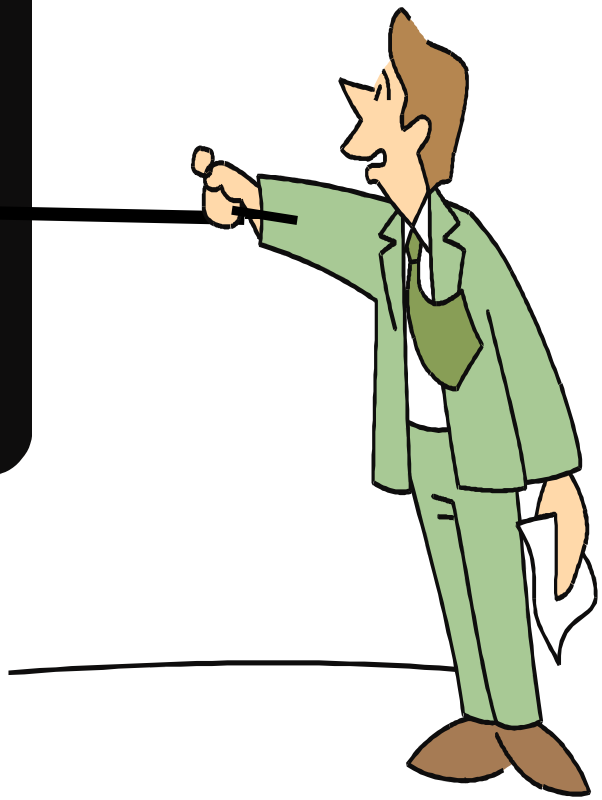
Introduction
organisation - STM32 ? - performance - marché

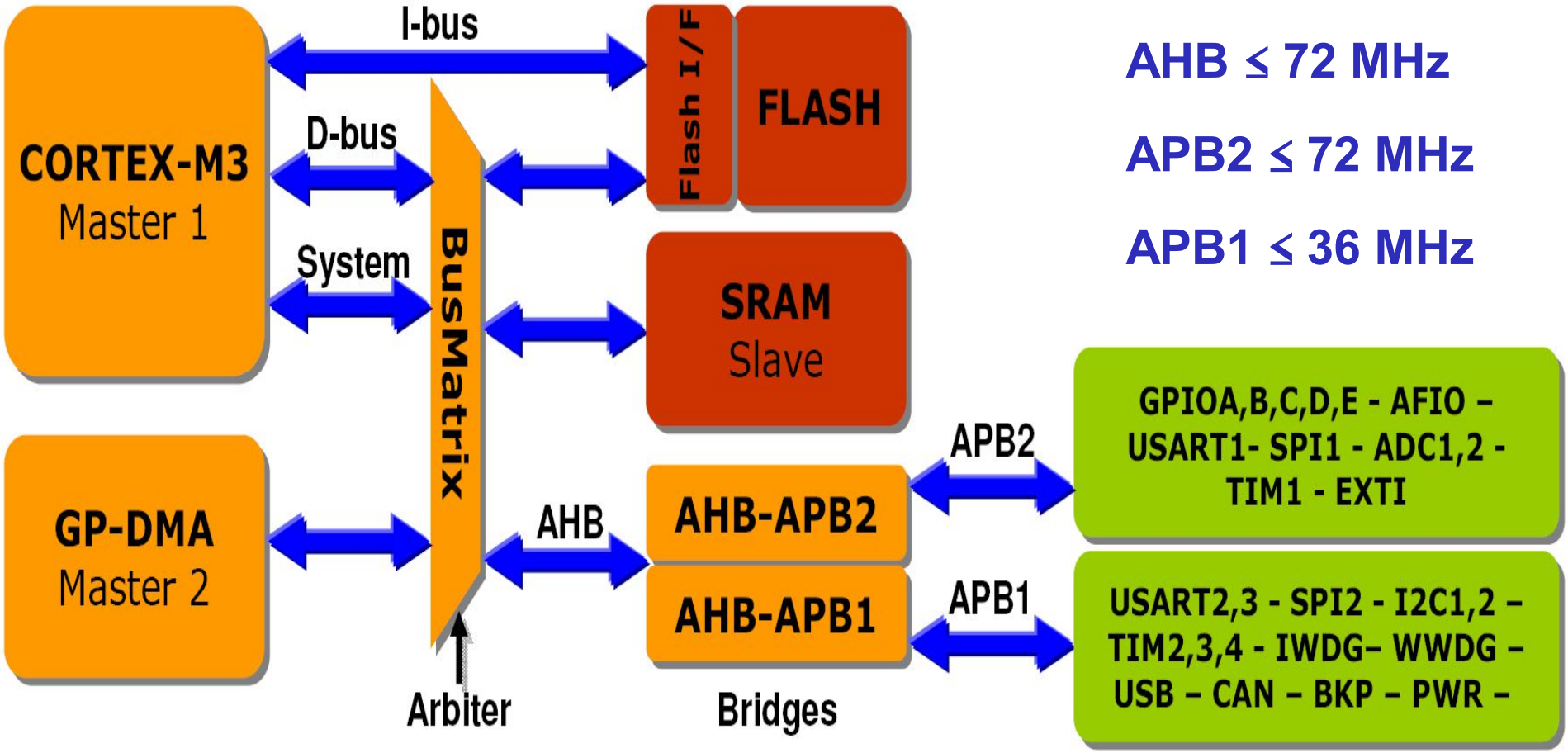
ARM-Cortex
cpu - adresses - données – interruptions

Outil Keil
cible – config – librairie

Périphériques
horloges - gpio - dma - adc – timers

Bus Can
Intérêt - protocole – erreurs – timing – stm32 - application



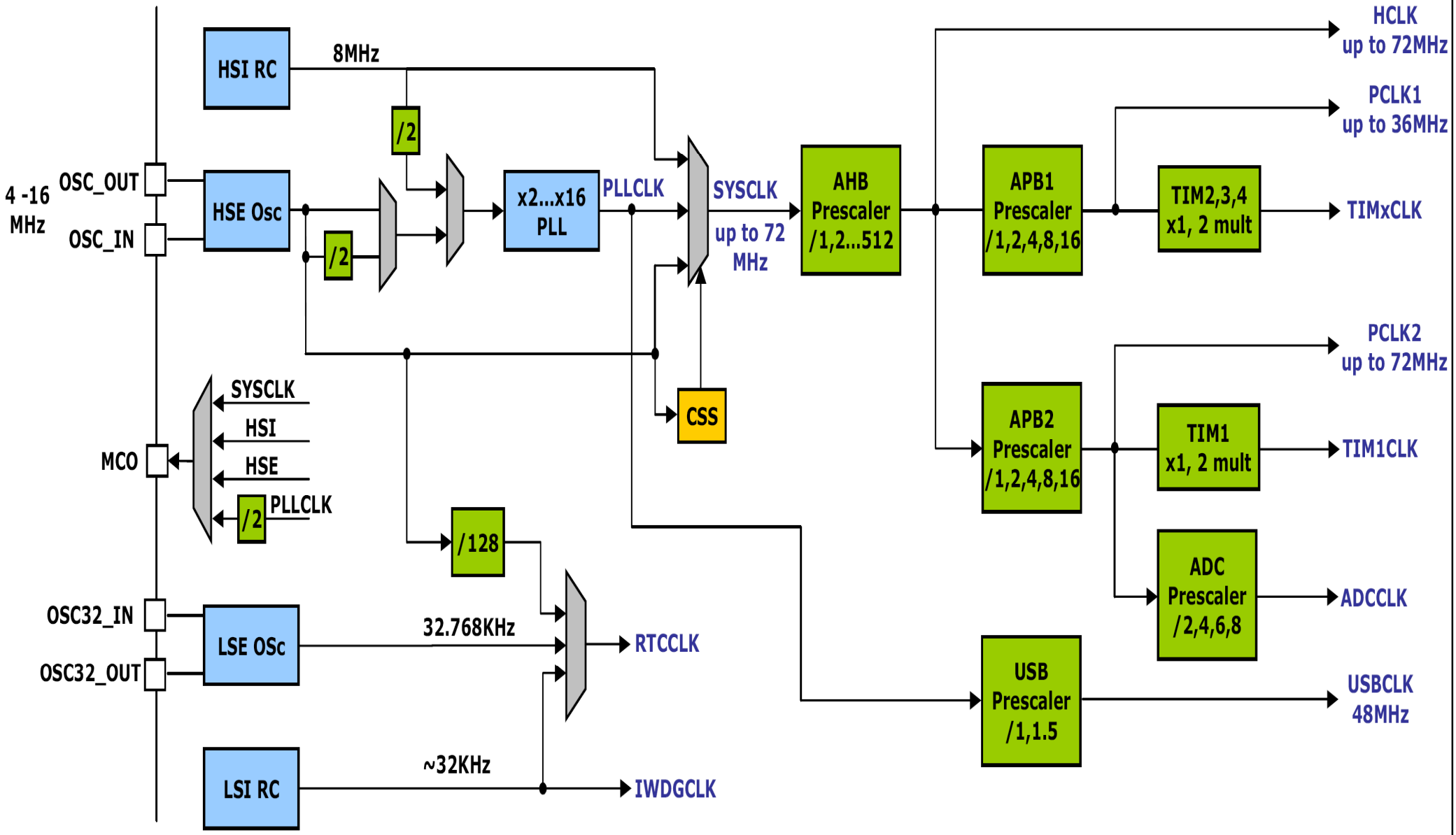


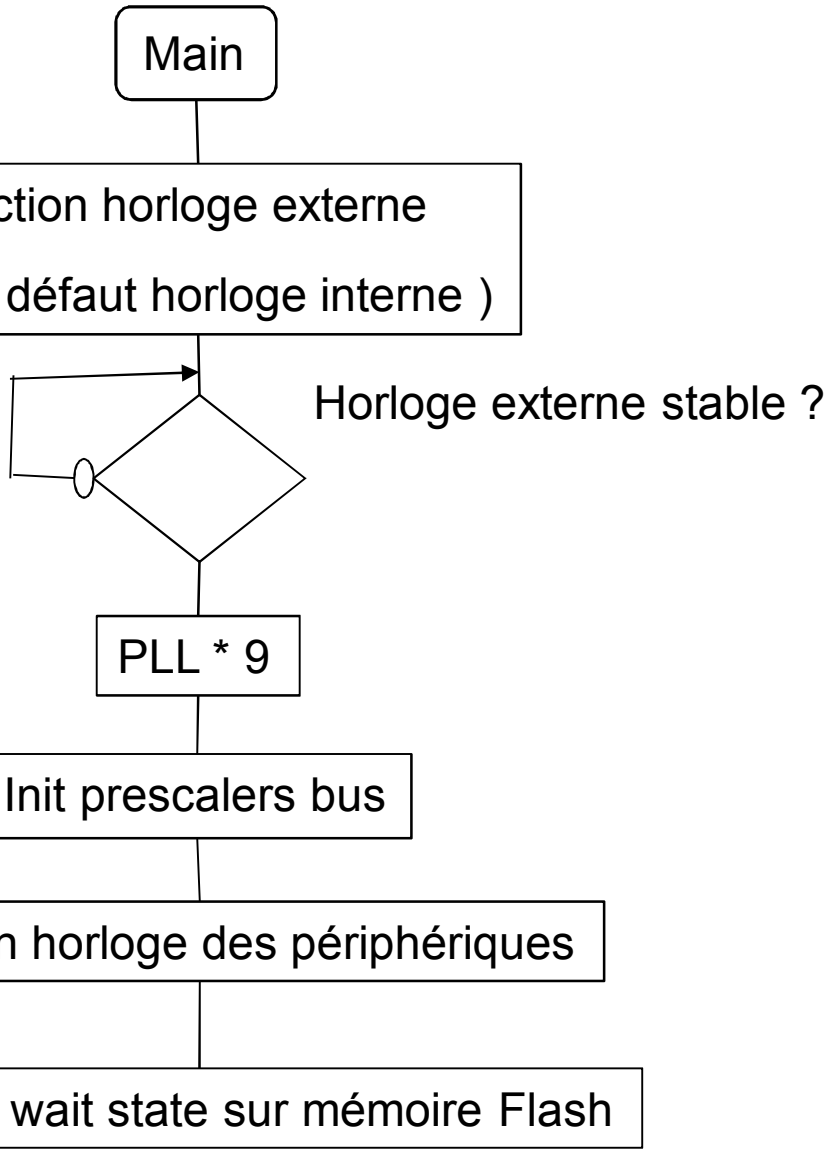
AHB ≤ 72 MHz

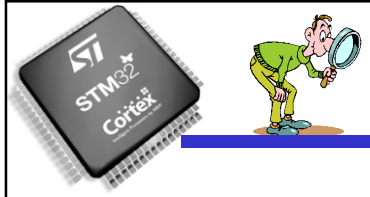
APB2 ≤ 72 MHz

APB1 ≤ 36 MHz

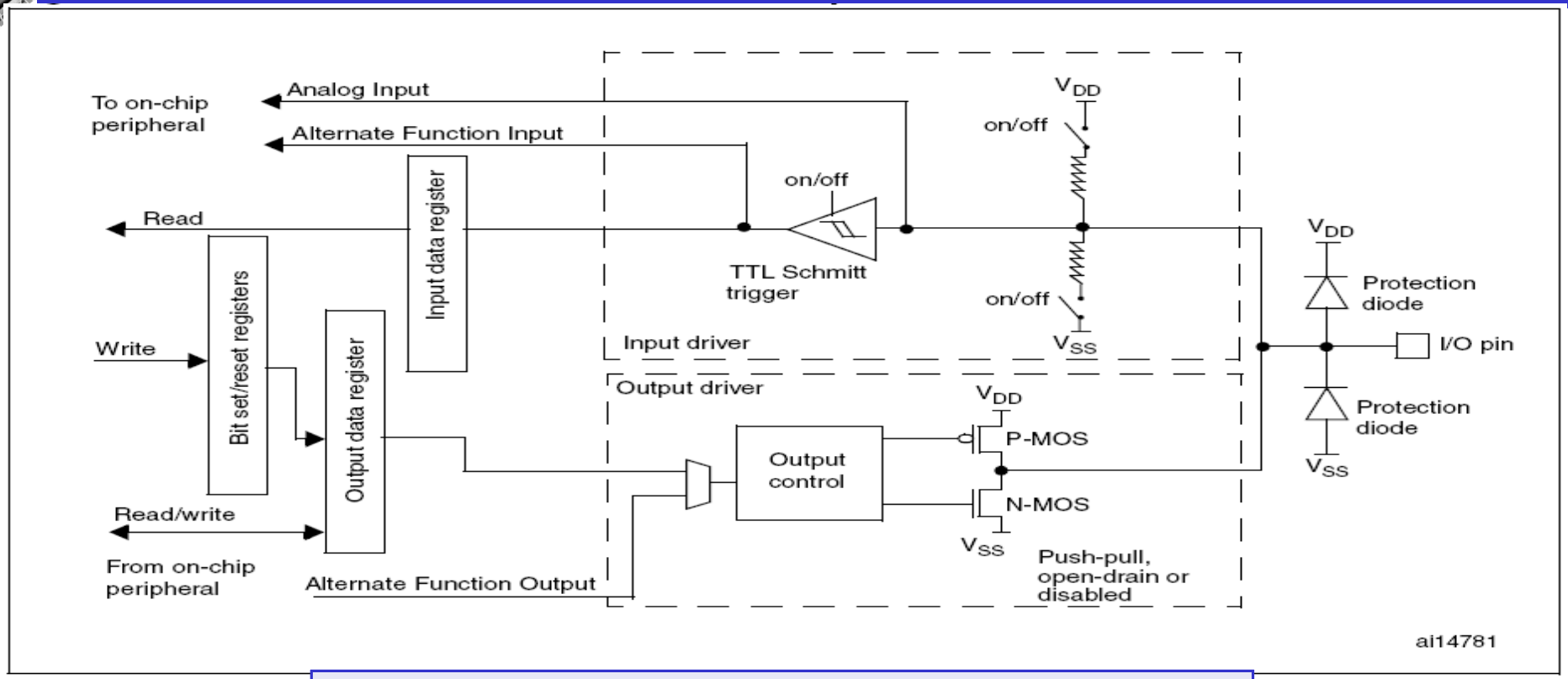
18 MHz SPI, 4,5 Mbps USART, 72 MHz PWM, 18 MHz GPIO


HSI RC
8MHz
/2
4 -16 MHz
OSC_OUT
OSC_IN
HSE Osc
/2
x2...x16 PLL
PLLCLK
SYSCLK
up to 72 MHz
AHB Prescaler /1,2...512
APB1 Prescaler /1,2,4,8,16
APB2 Prescaler /1,2,4,8,16
HCLK up to 72MHz
PCLK1 up to 36MHz
PCLK2 up to 72MHz
TIM2,3,4 x1, 2 mult
TIMxCLK
TIM1 x1, 2 mult
TIM1CLK
ADC Prescaler /2,4,6,8
ADCCLK
CSS
SYSCLK
HSI
HSE
PLLCLK
/2
MCO
/128
OSC32_IN
OSC32_OUT
LSE OSc
32.768KHz
RTCCLK
LSI RC
~32KHz
IWDGCLK
USB Prescaler /1,1.5
USBCLK 48MHz

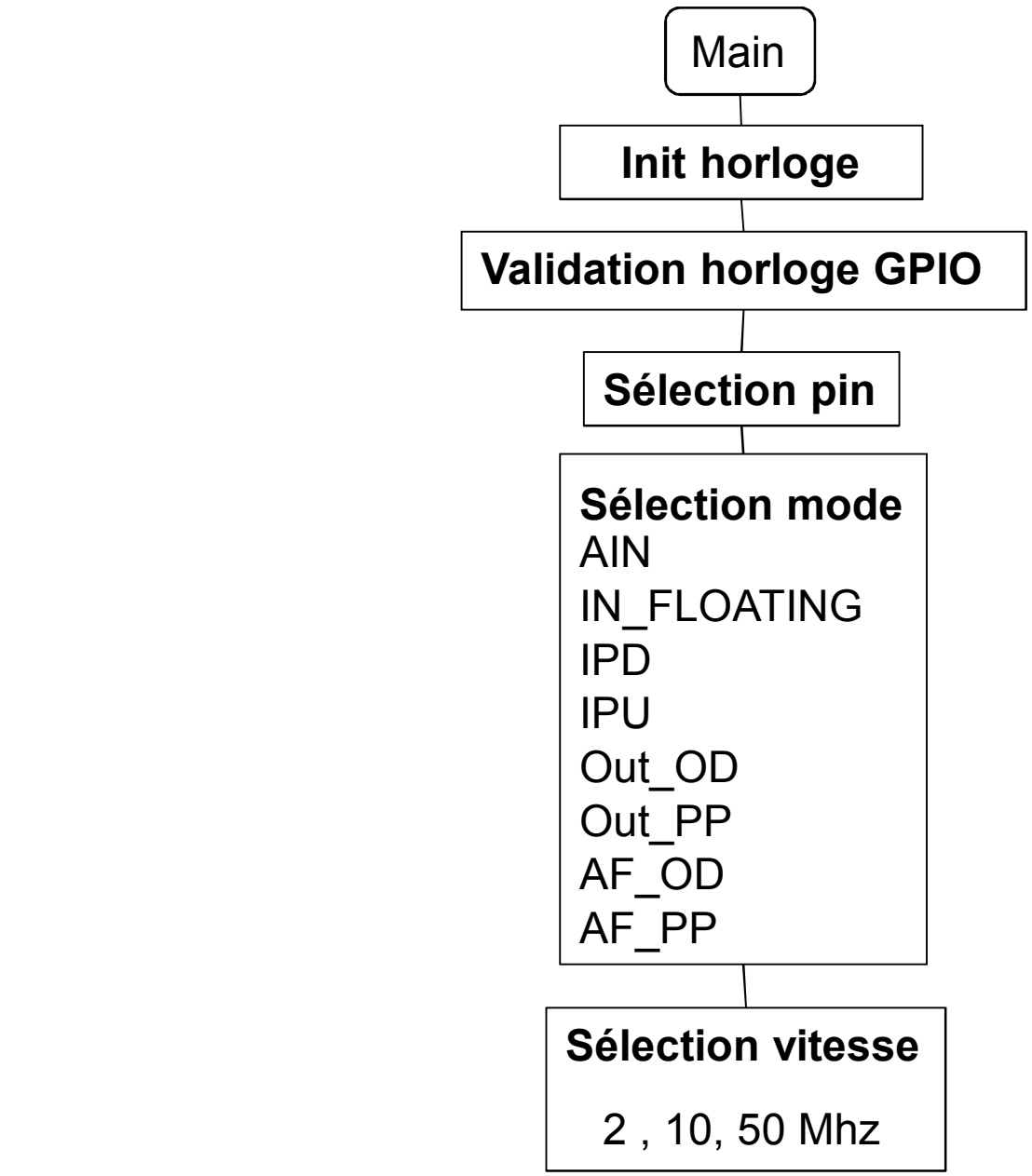

Main
Sélection horloge externe
(par défaut horloge interne)
Horloge externe stable ?
PLL * 9
Init prescalers bus
Validation horloge des périphériques
Insertion 2 wait state sur mémoire Flash




```
ErrorStatus HSEStartUpStatus;

 RCC_HSEConfig(RCC_HSE_ON);
 HSEStartUpStatus = RCC_WaitForHSEStartUp();                    /* Wait till HSE is ready */
 if(HSEStartUpStatus == SUCCESS)          {
     RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
     RCC_PLLCmd(ENABLE);
     RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
     RCC_HCLKConfig(RCC_SYSCLK_Div1);
     RCC_PCLK1Config(RCC_HCLK_Div2);
     RCC_PCLK2Config(RCC_HCLK_Div1);
     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_SRAM|RCC_AHBPeriph_FLITF, ENABLE );
 } else  while (1);

FLASH_SetLatency(FLASH_Latency_2);
FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
```

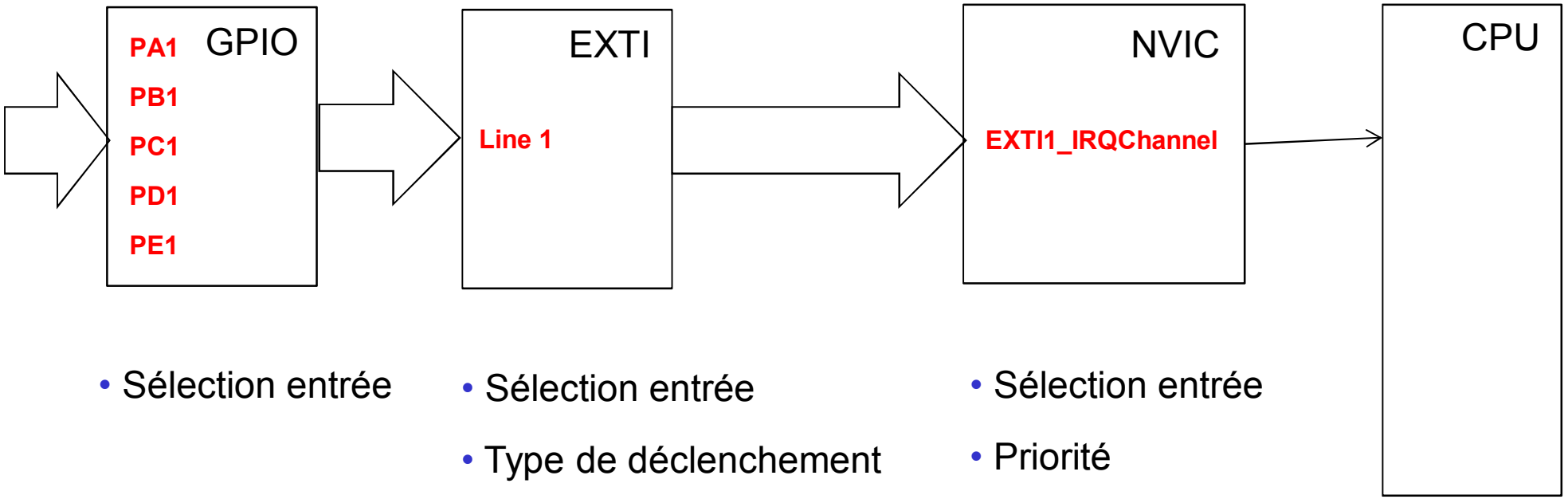


CONFIGURATION MODE		
General purpose output	Push-pull	Max output speed 10 Mhz Max output speed 20 Mhz Max output speed 50 Mhz
	Open-drain	
Alternate Function	Push-pull	
	Open-drain	
Input	Analog input	N/A
	Input floating	
	Input pull-down	
	Input pull-up	


Main
Init horloge
Validation horloge GPIO
Sélection pin
Sélection mode
AIN
IN_FLOATING
IPD
IPU
Out_OD
Out_PP
AF_OD
AF_PP
Sélection vitesse
2 , 10, 50 Mhz

```
void GPIO_init() {
  GPIO_InitTypeDef    GPIO_InitStructure;

  GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_8;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
}

/* Toggle GPIO_LED pin 8 */
  GPIO_WriteBit(GPIOB, GPIO_Pin_8, (BitAction)(1^GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_8)));
```

✓ 19 EXTI dont 16 E/S



- Sélection entrée

- Sélection entrée
- Type de déclenchement
- Mode
- Validation

- Sélection entrée
- Priorité
- Validation

Main

Init horloge

Validation horloge GPIO, AFIO

Sélection entrée GPIO

Sélection EXTI line

Sélection EXTI Mode
Interrupt
Event

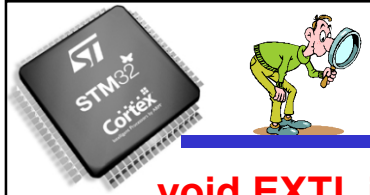
Sélection front de déclenchement
Falling
Rising
Rising_Falling

Validation EXTI

Entrée d'interruption NVIC

Priorité d'interruption
PriorityGroup
PreemptionPriority
SubPriority

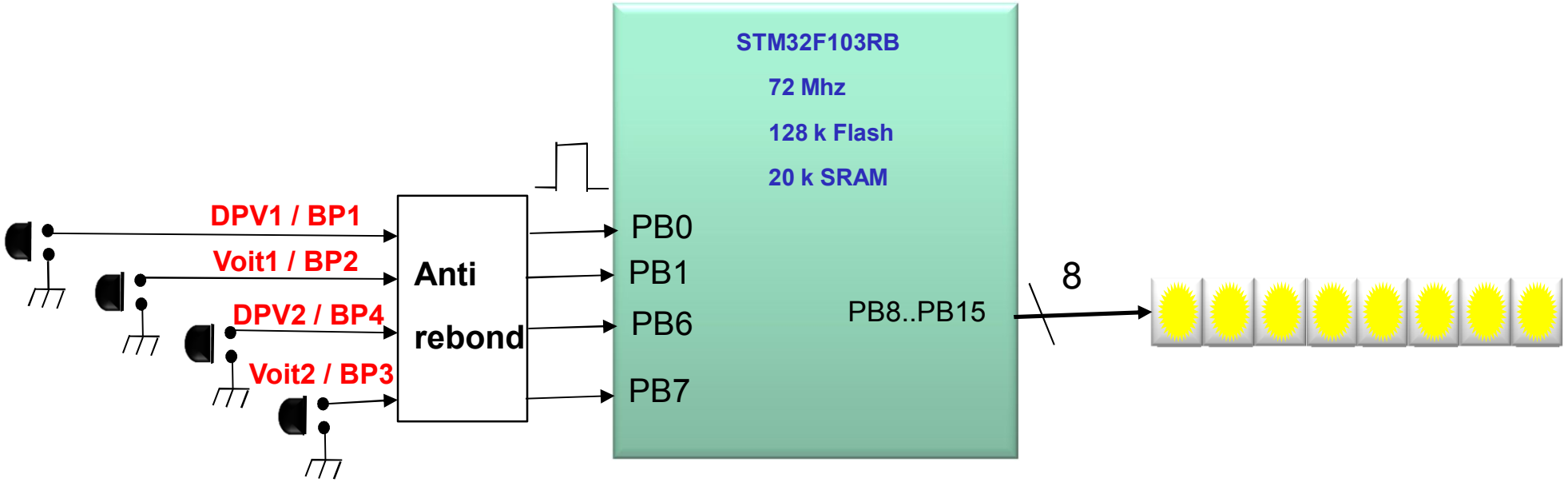
Validation NVIC




```
void EXTI_init () {
    EXTI_InitTypeDef   EXTI_InitStructure;
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource1); /* Selects PA.1 as EXTI */
    EXTI_InitStructure.EXTI_Line = EXTI_Line1;                  /*  Line 1 */
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);                             }

void NVIC_init(void)   {
    NVIC_InitTypeDef   NVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); /* Configure 2 bits for preemption priority */
    NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQChannel; /* Enable the EXTI1 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);                             }
```

- Combien de ports possèdent le STM32 ?
- Quels sont les configurations possibles des entrées / sorties ?
- Quel est le rôle du Clock Security System ?
- Combien d'interruptions externes gère le cortex M3?
- Quelles interruptions peuvent réveiller le microcontrôleur ?
- Quel est le maximum des fréquences des bus AHB, APB1, APB2 ?
- Quelle est la fréquence du quartz sur le kit MCBSTM32?
- Quel est le rôle de la PLL interne? Intérêt ?
- Rôle du périphérique RTC?

```
*-----------------------------------------------------------------------------------------------------
 * FONCTION: Commutation led PB8 à chaque front descendant sur le bouton
*                  poussoir BP4.
 *                 Utiliser une interruption avec la priorité la plus forte
*                  ( 2 bit de priorité et 2 bit de sous-priorité )
 *-----------------------------------------------------------------------------------------------------
```



*----------

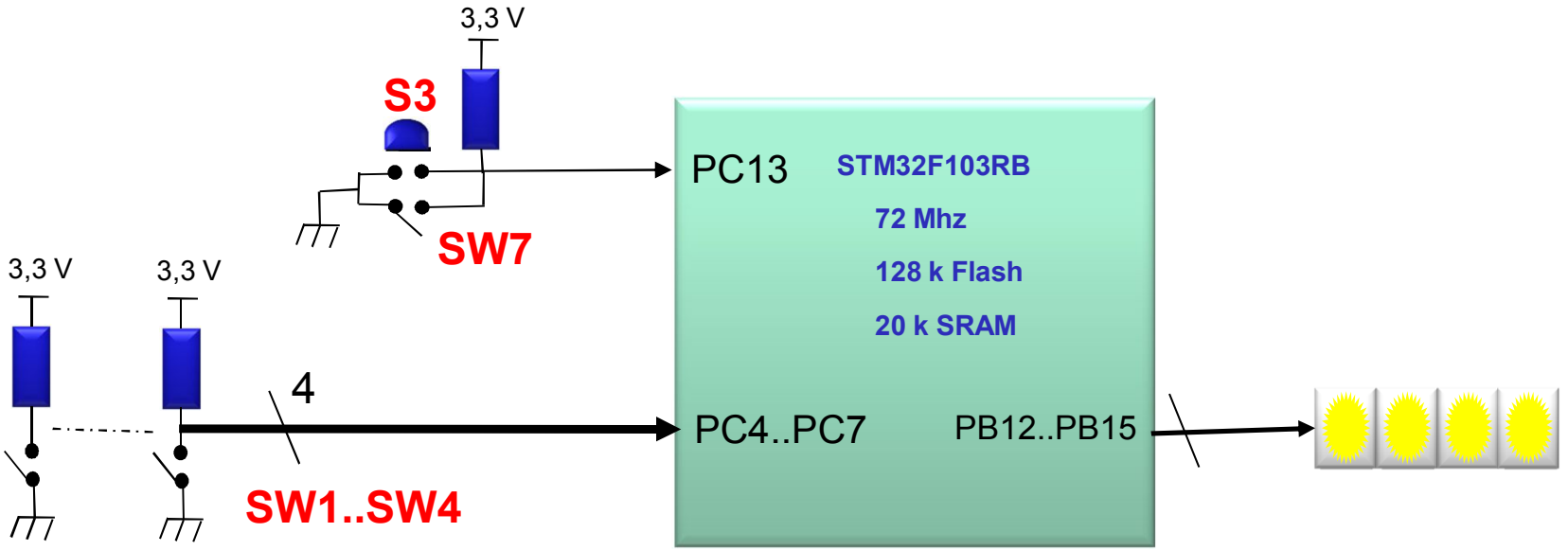
* FONCTION: Calculateur Multiplication

* Acquisition 1er Opérande (switches SW1 à SW4) via une interruption

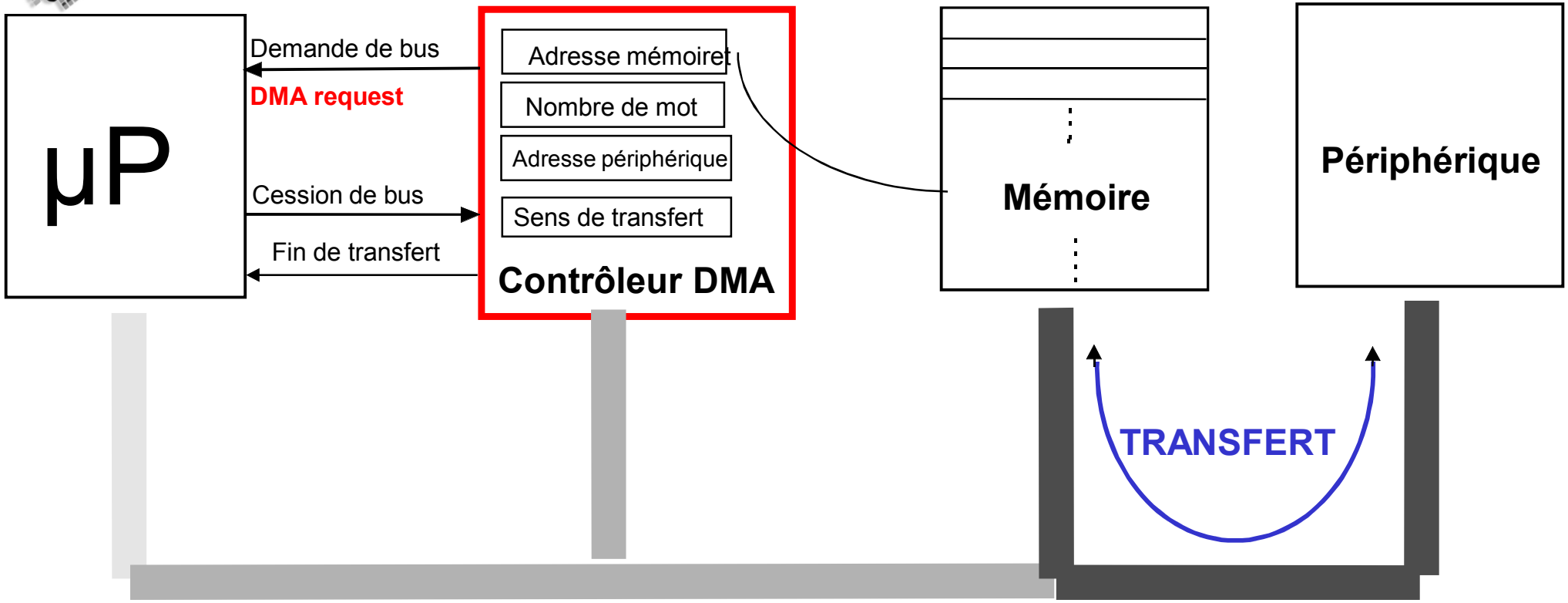
* Acquisition 2ème Opérande via l'interruption une seconde fois

* Affichage du résultat sur les 8 leds.

*----------

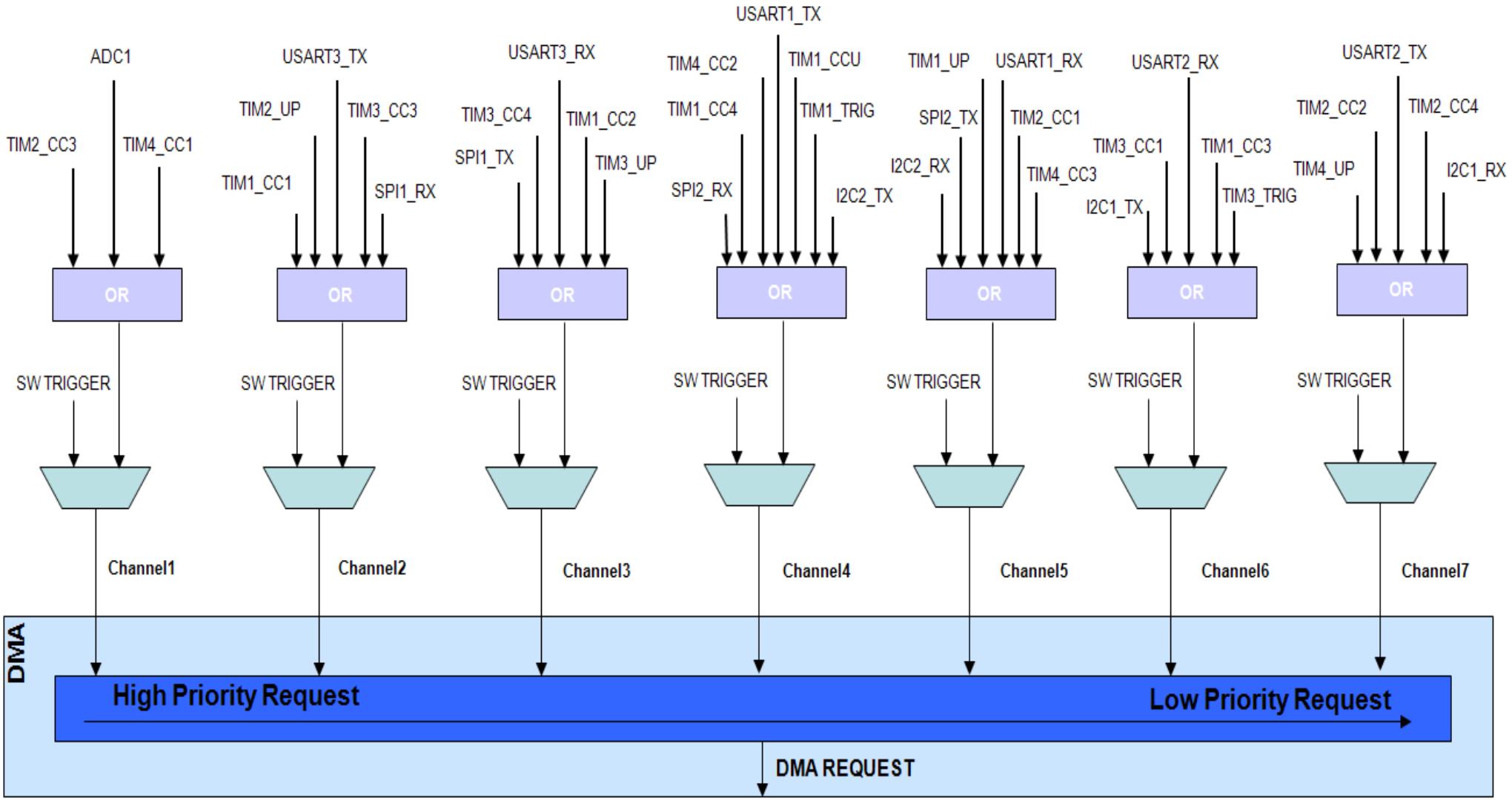


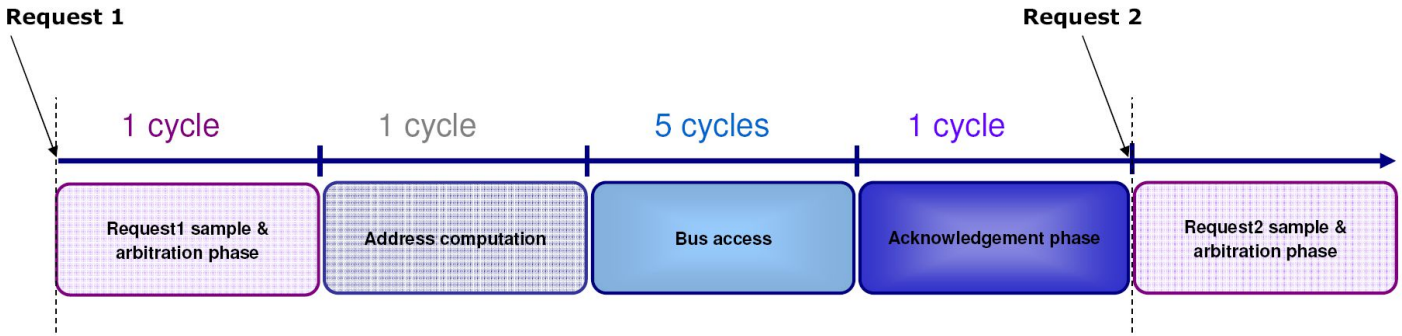
PRINCIPE

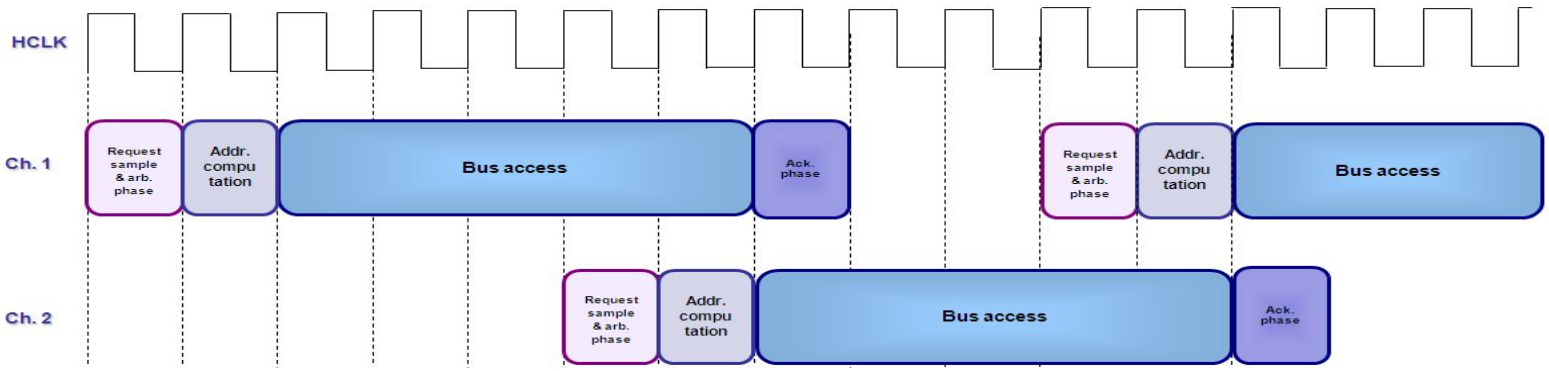


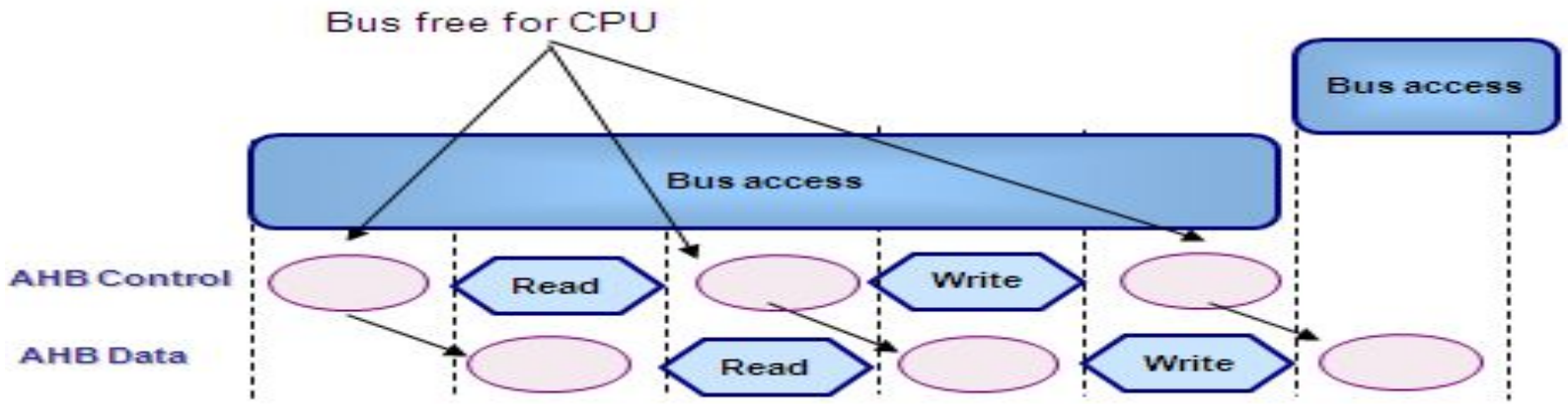
- ✓ **Mémoire / Mémoire**
- ✓ **Mémoire / Périphérique**
- ✓ **Périphérique / Mémoire**
- ✓ **Périphérique / Périphérique**

CANAUX DMA

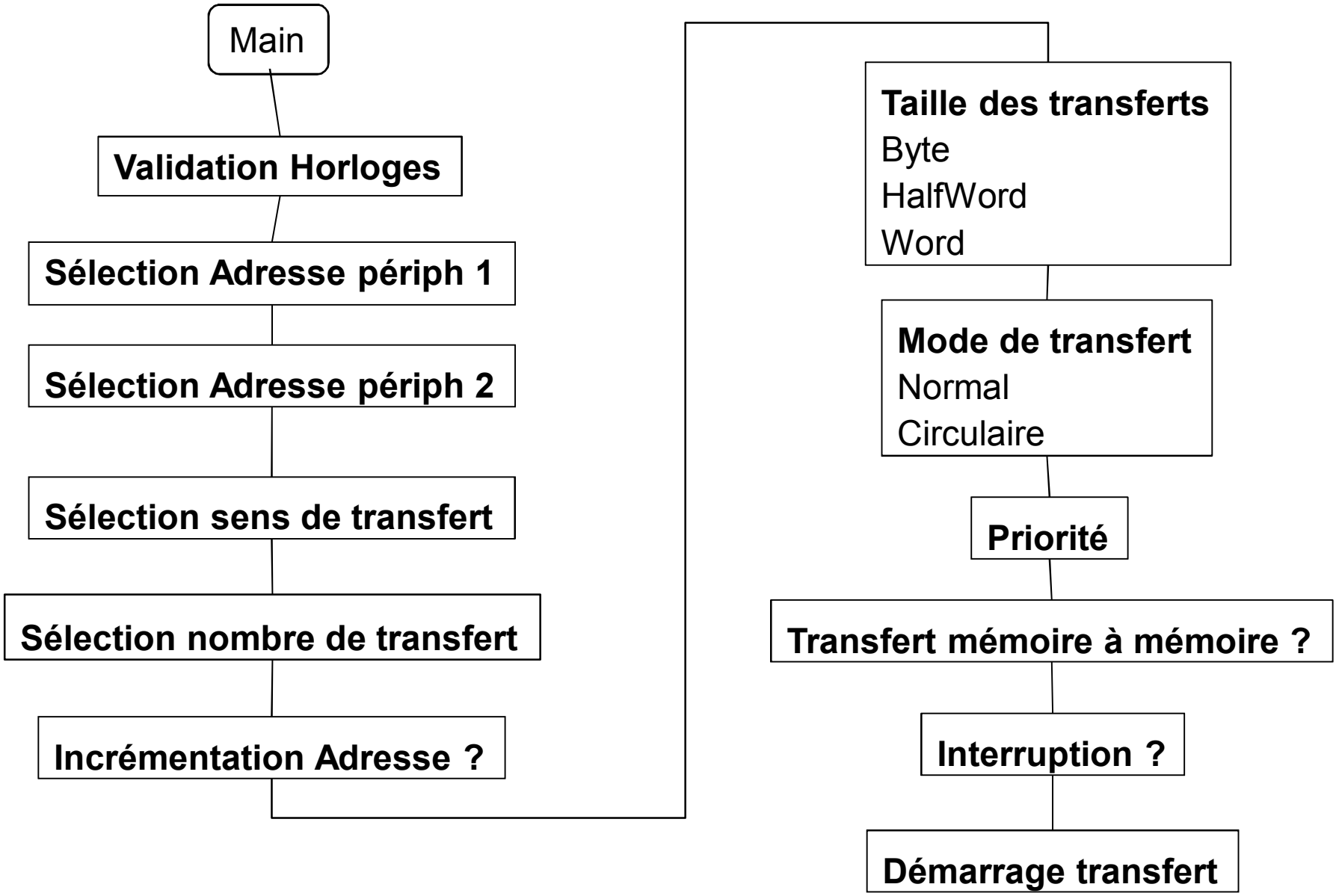

ADC1
TIM2_CC3
TIM4_CC1
USART3_TX
TIM2_UP
TIM3_CC3
TIM1_CC1
SPI1_RX
USART3_RX
TIM3_CC4
TIM1_CC2
SPI1_TX
TIM3_UP
USART1_TX
TIM4_CC2
TIM1_CCU
TIM1_CC4
TIM1_TRIG
SPI2_RX
I2C2_TX
TIM1_UP
USART1_RX
SPI2_TX
TIM2_CC1
I2C2_RX
TIM4_CC3
USART2_RX
TIM3_CC1
TIM1_CC3
I2C1_TX
TIM3_TRIG
USART2_TX
TIM2_CC2
TIM2_CC4
TIM4_UP
I2C1_RX
OR
SW TRIGGER
Channel1
Channel2
Channel3
Channel4
Channel5
Channel6
Channel7
DMA
High Priority Request
Low Priority Request
DMA REQUEST


Request 1
Request 2
1 cycle
1 cycle
5 cycles
1 cycle
Request1 sample & arbitration phase
Address computation
Bus access
Acknowledgement phase
Request2 sample & arbitration phase


HCLK
Ch. 1
Ch. 2
Request sample & arb. phase
Addr. compu tation
Bus access
Ack. phase


Bus free for CPU
Bus access
Bus access
AHB Control
AHB Data
Read
Write
Read
Write

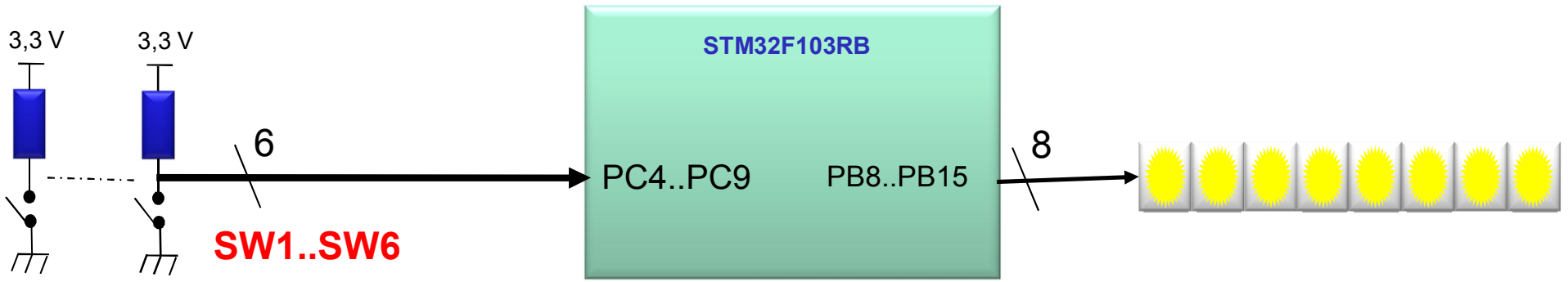
ORGANIGRAMME



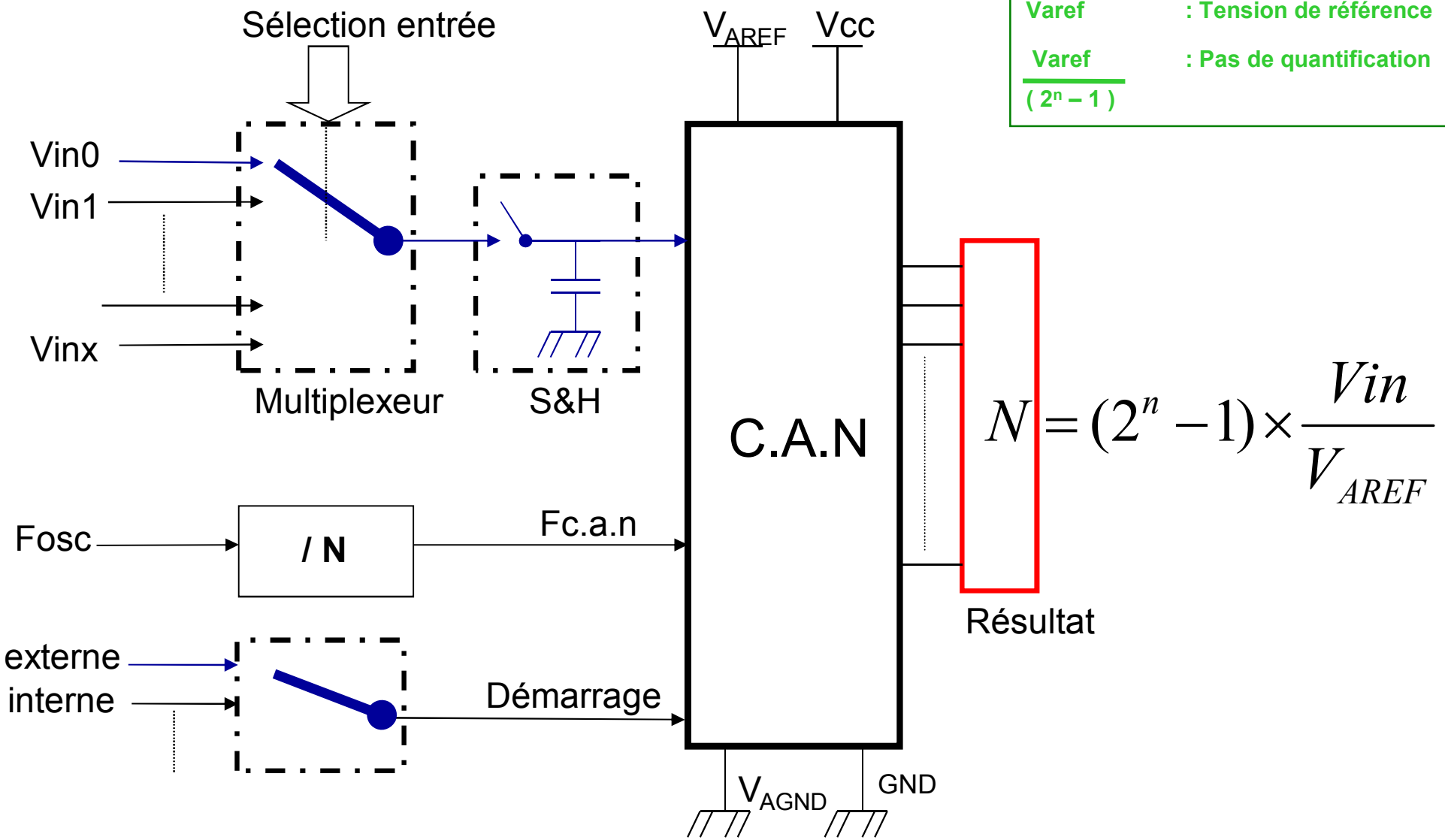
```
DMA_InitTypeDef    DMA_InitStruct;

DMA_InitStruct.DMA_PeripheralBaseAddr = (u32)&message1;
DMA_InitStruct.DMA_MemoryBaseAddr = (u32)&message2;
DMA_InitStruct.DMA_DIR = DMA_DIR_PeripheralSRC;
DMA_InitStruct.DMA_BufferSize = 10000;
DMA_InitStruct.DMA_PeripheralInc = DMA_PeripheralInc_Enable;
DMA_InitStruct.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_InitStruct.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_InitStruct.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_InitStruct.DMA_Mode = DMA_Mode_Normal;
DMA_InitStruct.DMA_Priority  = DMA_Priority_Medium;
DMA_InitStruct.DMA_M2M = DMA_M2M_Enable;
DMA_Init( DMA1_Channel1, &DMA_InitStruct );
DMA_ITConfig(DMA1_Channel1, DMA_IT_TC, ENABLE);
DMA_Cmd(DMA1_Channel1, ENABLE );
```

- Combien de voie DMA possède le STM32 ?
- Lister les périphériques qui ont une interface DMA ?
- Quelles interruptions chaque voie peut générer ?
- Quelles voies peuvent déclencher un transfert de mémoire à mémoire ?
- Quel est l'intérêt d'un circuit DMA ?
- Pourquoi le DMA n'utilise t-il que 40 % de la bande passante du bus ?
- Quelle taille mémoire maximum peut transférer le DMA en mode Normal ?
- Intérêt du mode circulaire ?
- Sur quel bus est connecté le DMA ?

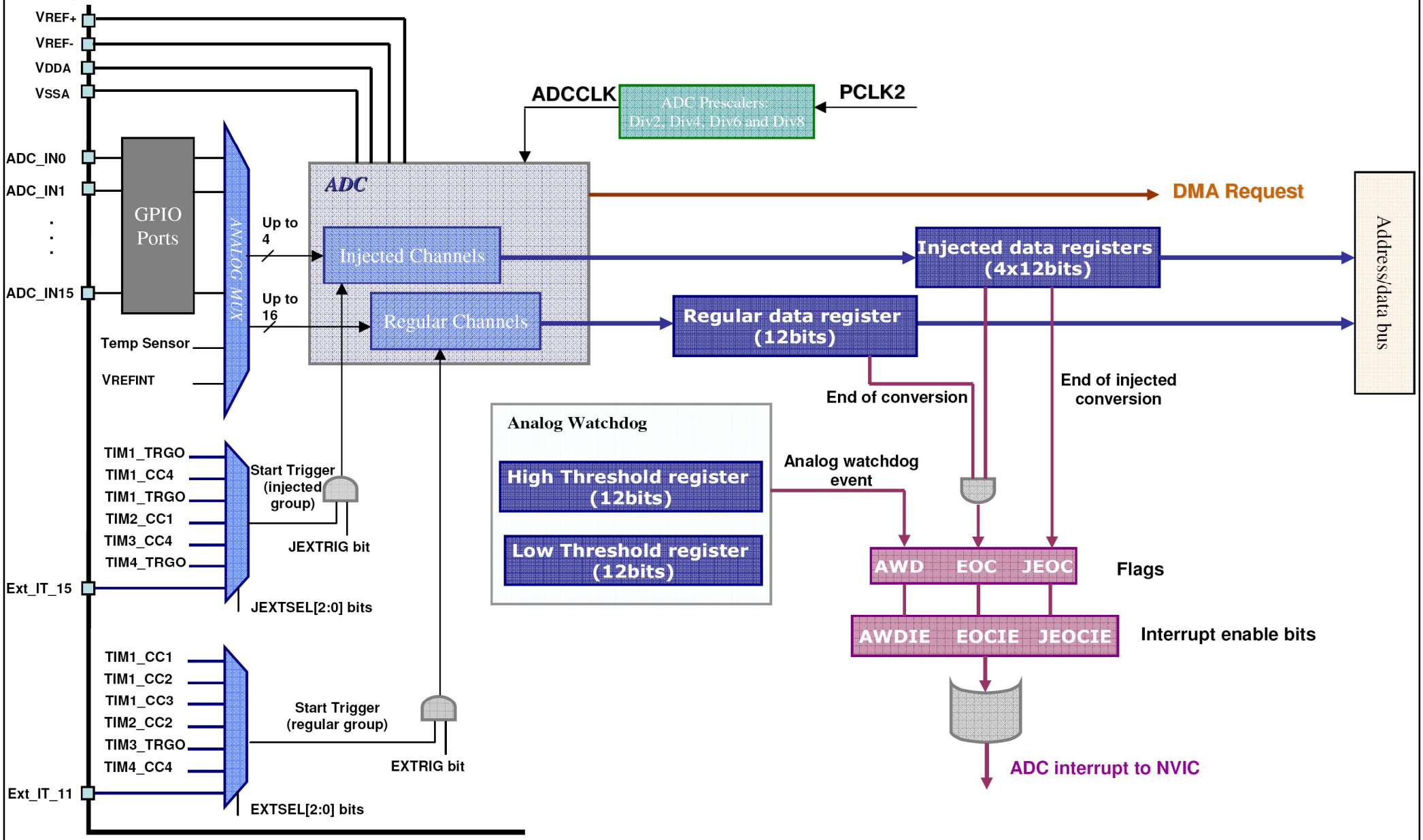
```
*-----------------------------------------------------------------------------------------------------------------------
* FONCTION: Transfert mémoire Flash vers mémoire SRAM sans et avec DMA du message suivant:
*
*           uc8 message1[10000]=" Centre de microelectronique de provence - George Charpak";
*           u8 message2[10000];
*
*  Mesurer le temps de transfert DMA en utilisant l'interruption DMA fin transfert, comparer
*-----------------------------------------------------------------------------------------------------------------------
*-----------------------------------------------------------------------------------------------------------------------
* FONCTION: Transférer en permanence  les Dip-switches ( SW1 à SW4 ) sur les Leds via le DMA
*-----------------------------------------------------------------------------------------------------------------------
```



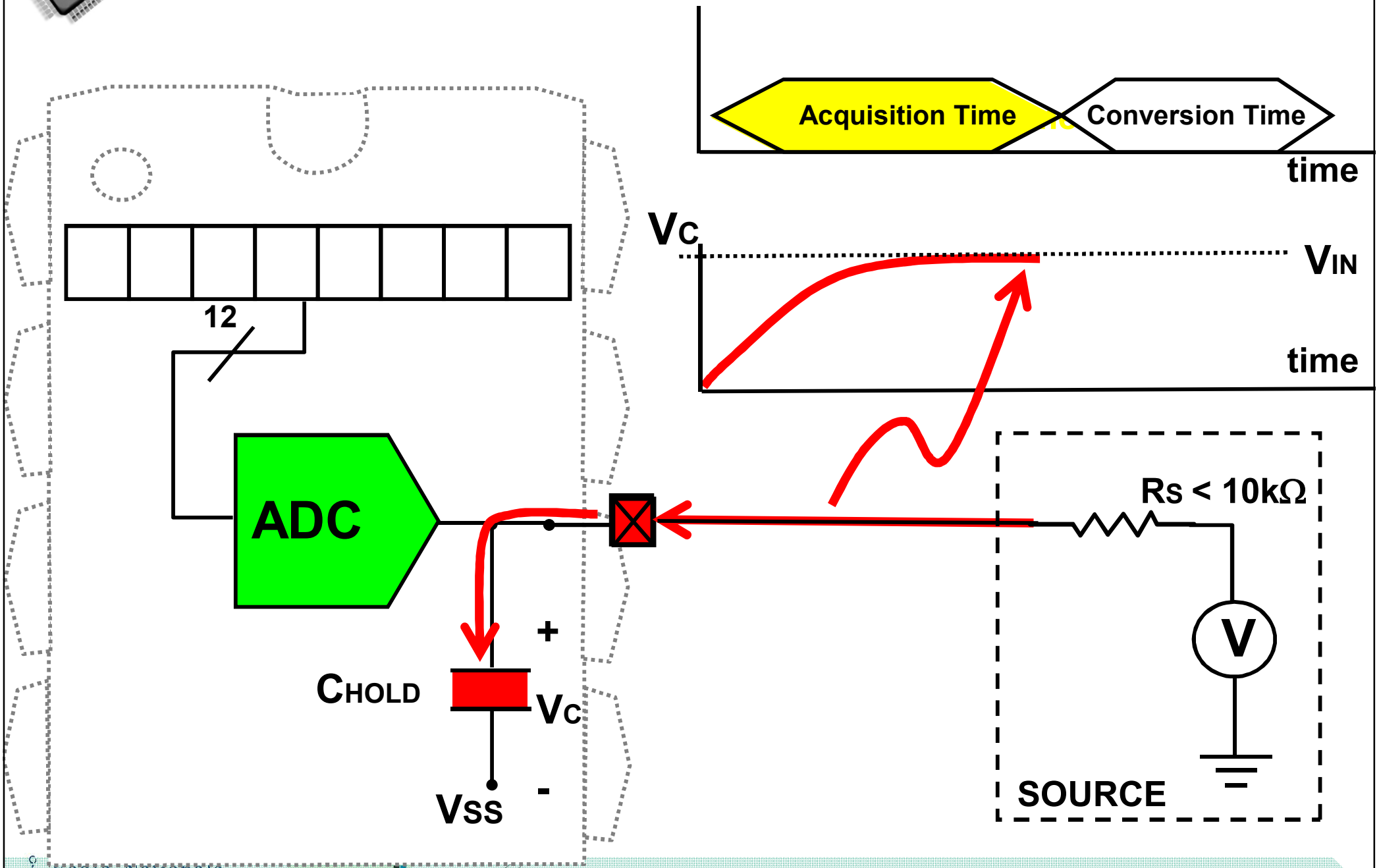
n	: Résolution
Vin	: Voie analogique
Varef	: Tension de référence
Varef / ($2^n - 1$)	: Pas de quantification



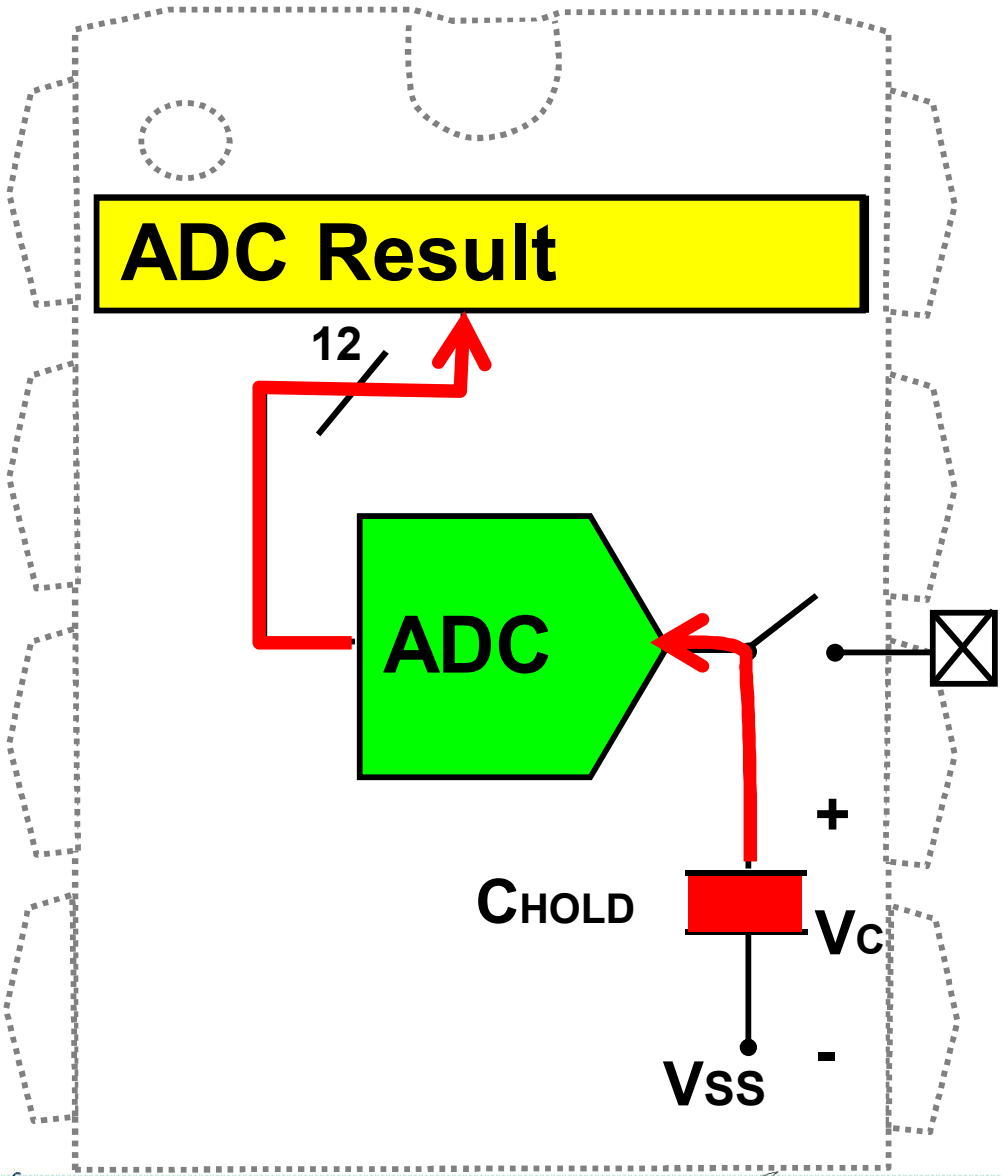
$$N = (2^n - 1) \times \frac{Vin}{V_{AREF}}$$

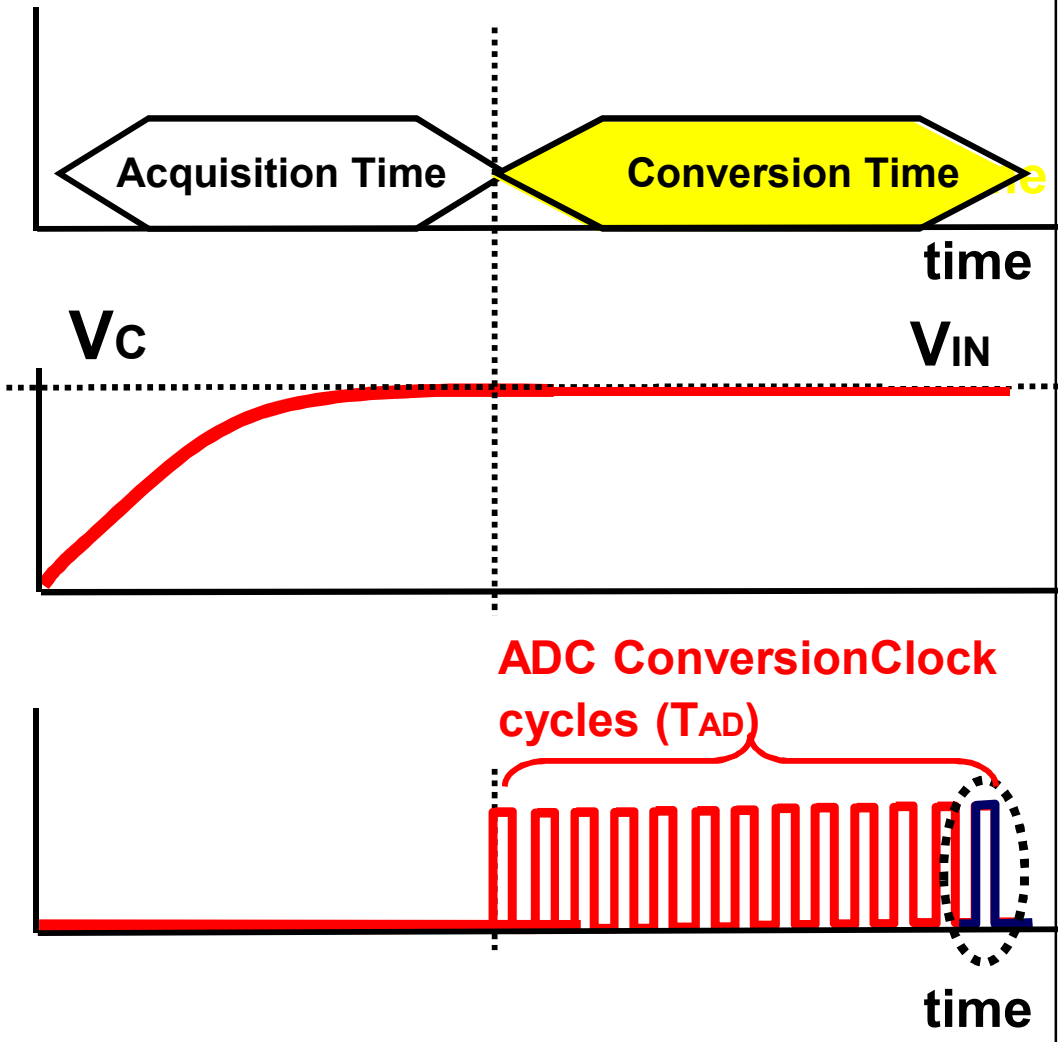

VREF+
VREF-
VDDA
VSSA
ADCCLK
ADC Prescalers: Div2, Div4, Div6 and Div8
PCLK2
ADC_IN0
ADC_IN1
ADC_IN15
GPIO Ports
ANALOG MUX
Up to 4
Up to 16
Temp Sensor
VREFINT
ADC
Injected Channels
Regular Channels
DMA Request
Injected data registers (4x12bits)
Regular data register (12bits)
Address/data bus
End of conversion
End of injected conversion
Analog Watchdog
High Threshold register (12bits)
Low Threshold register (12bits)
Analog watchdog event
TIM1_TRGO
TIM1_CC4
TIM1_TRGO
TIM2_CC1
TIM3_CC4
TIM4_TRGO
Ext_IT_15
Start Trigger (injected group)
JEXTRIG bit
JEXTSEL[2:0] bits
AWD
EOC
JEOC
Flags
AWDIE
EOCIE
JEOCIE
Interrupt enable bits
TIM1_CC1
TIM1_CC2
TIM1_CC3
TIM2_CC2
TIM3_TRGO
TIM4_CC4
Ext_IT_11
Start Trigger (regular group)
EXTRIG bit
EXTSEL[2:0] bits
ADC interrupt to NVIC

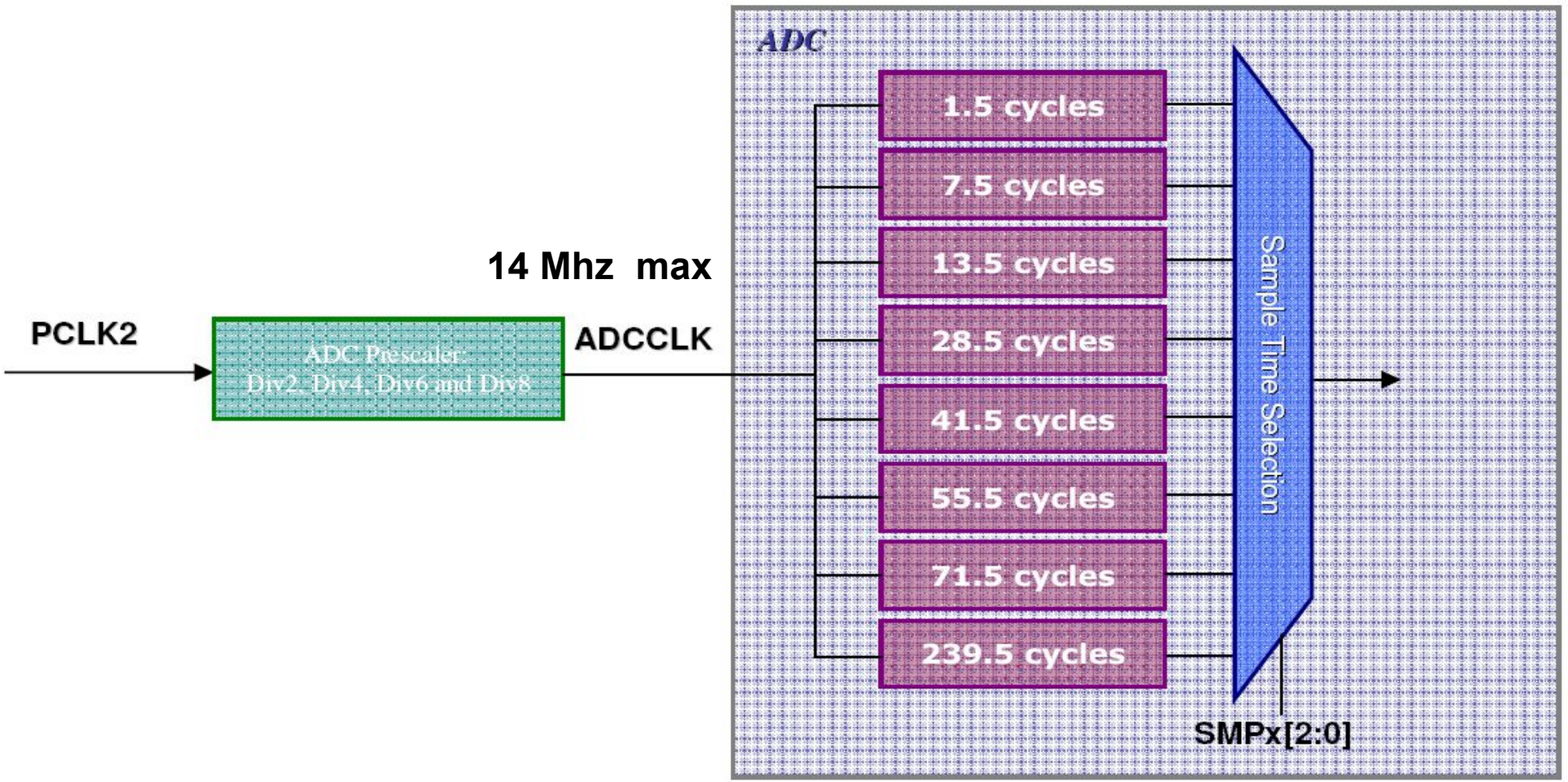
TEMPS D'ACQUISITION



TEMPS DE CONVERSION

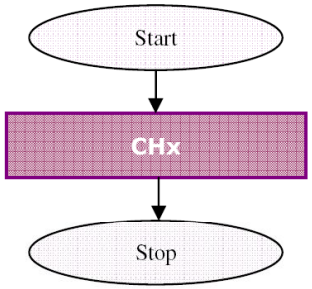






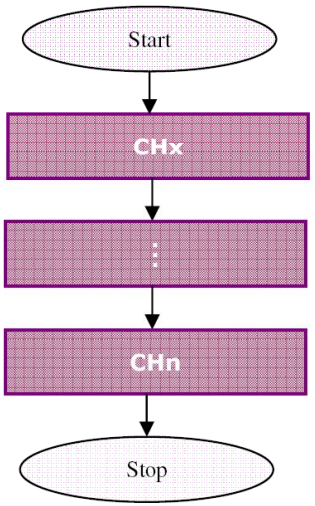
TOTAL CONVERSION = Temps d'acquisition + 12, 5 cycles = 1 µs max

MODE DE CONVERSION

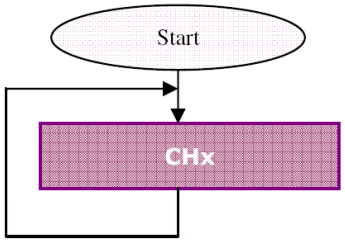


One channel

Single conversion mode

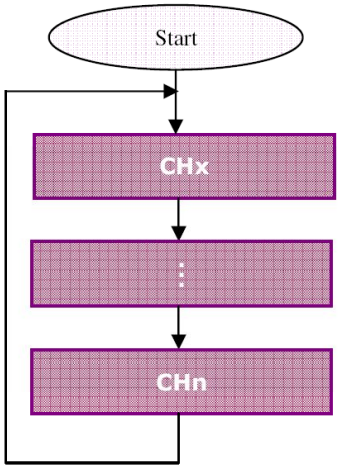


Multi-channels (Scan)
Single conversion mode

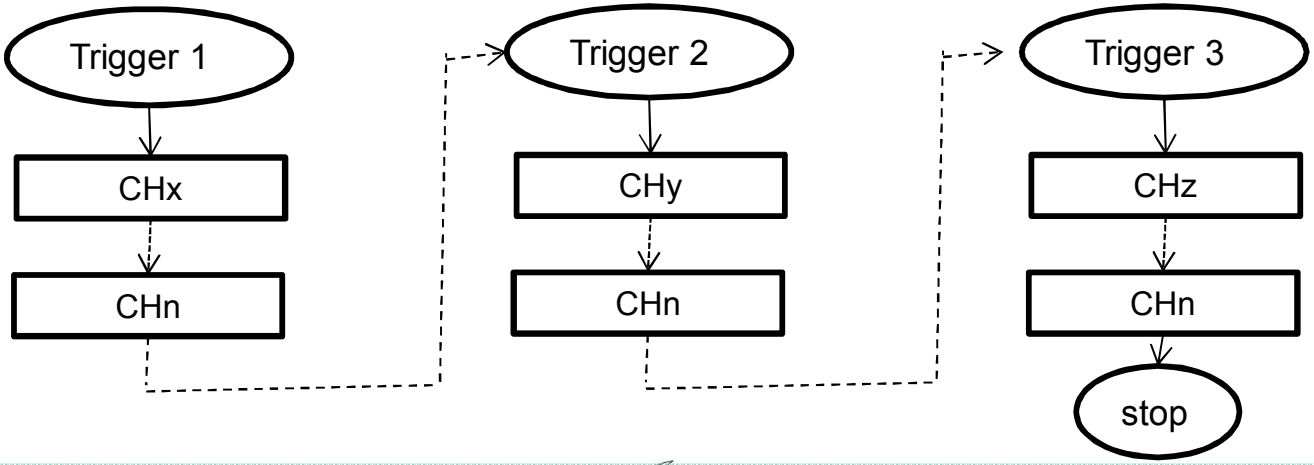


One channel

Continuous conversion mode



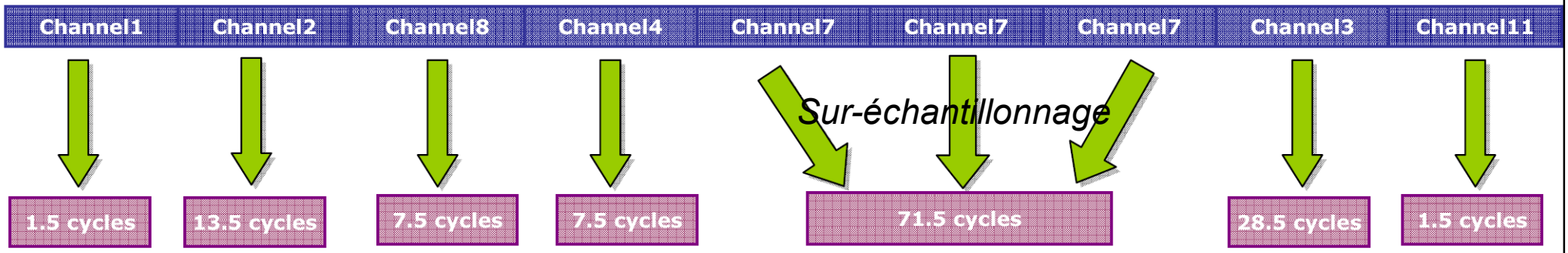
Multi-channels (Scan)
Continuous conversion mode



Discontinuous

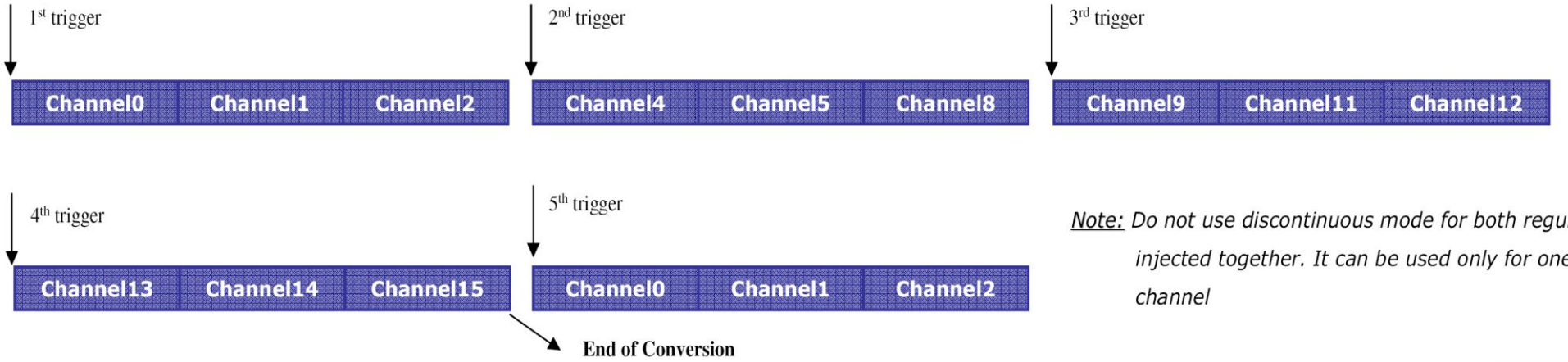
conversion mode

Exemple1: Conversion multi-channels 1, 2, 8, 4, 7, 3 et 11



Exemple2: Discontinuous mode, nombre de canaux 3

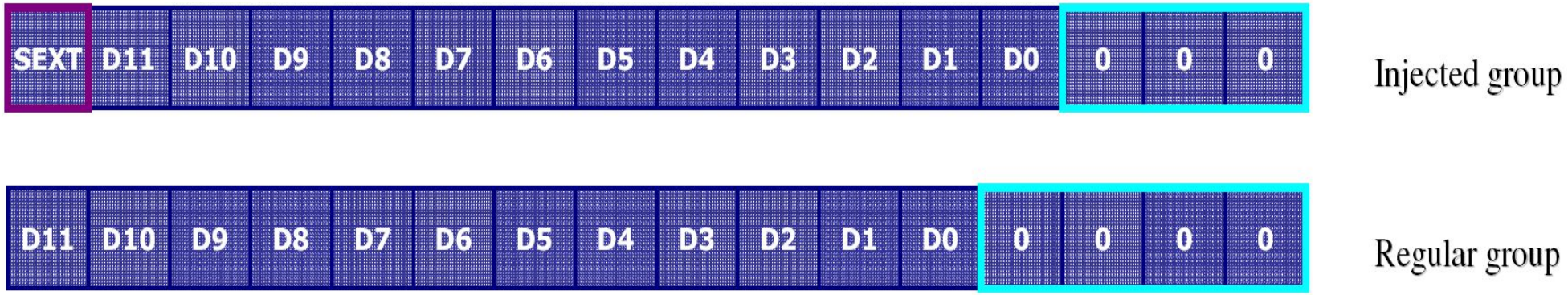
groupe de canaux: 0, 1, 2, 4, 5, 8, 9, 11, 12, 13, 14 et 15



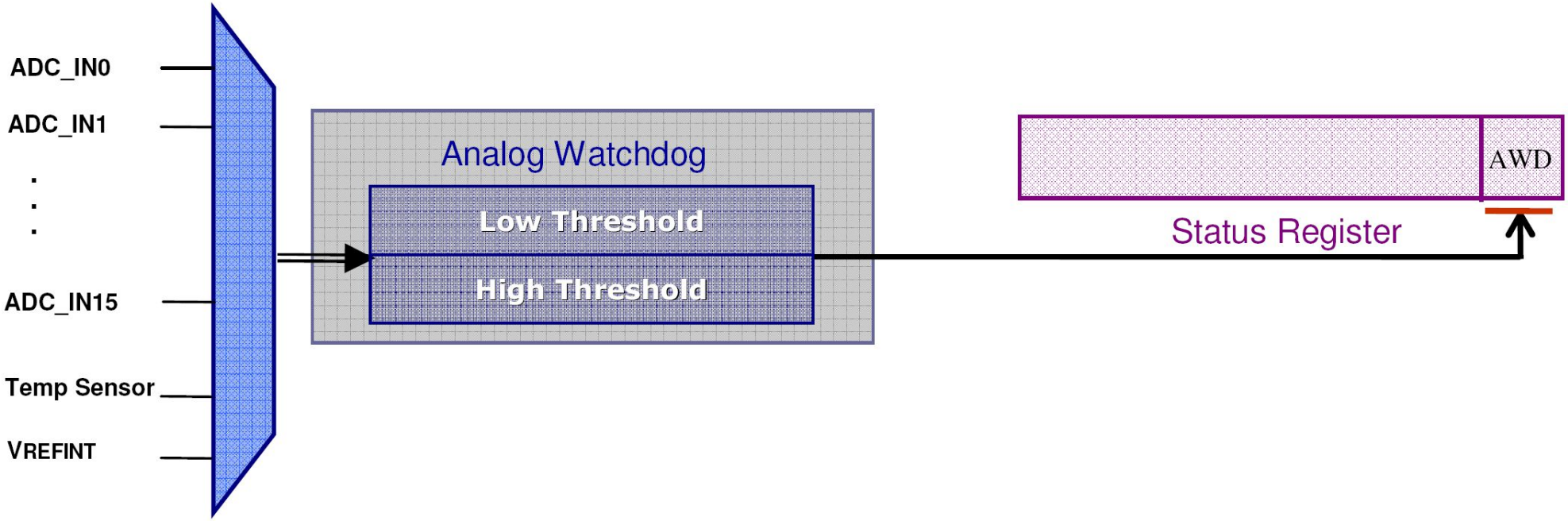
Note: Do not use discontinuous mode for both regular and injected together. It can be used only for one group channel

Right alignment

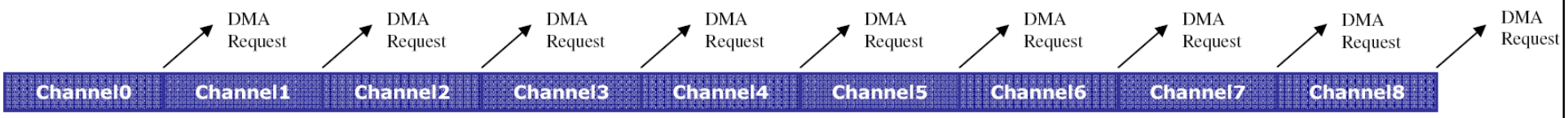
Left alignment

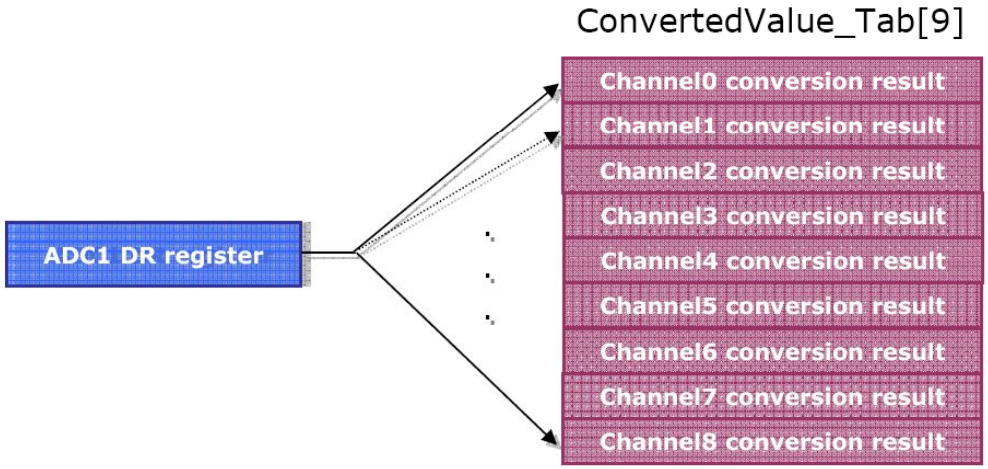


ANALOG WATCHDOG

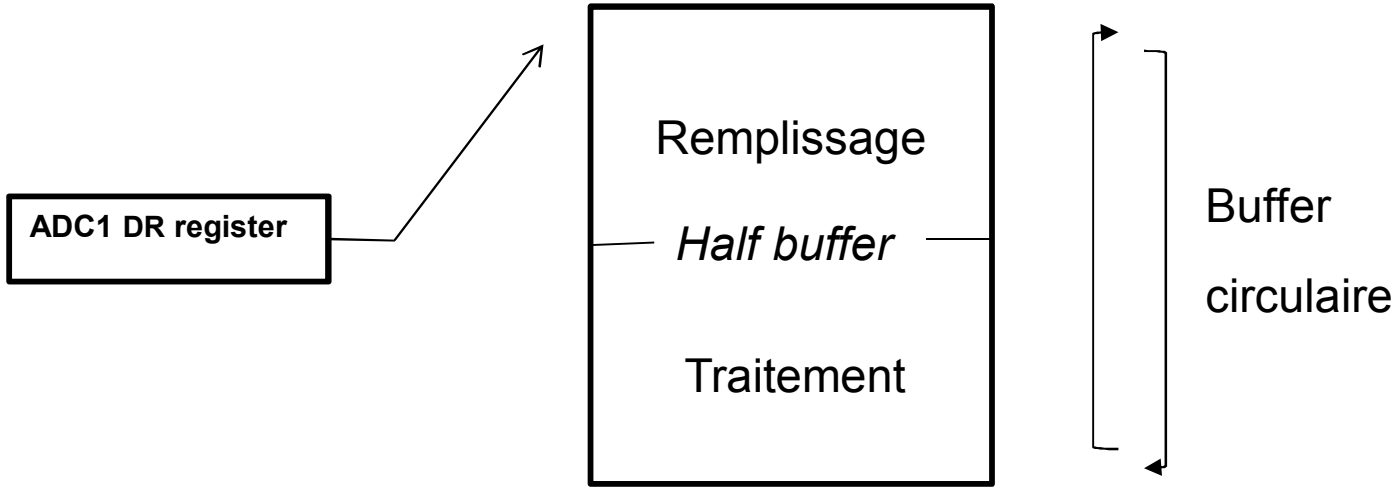


- ✓ **Seuil Haut et Bas: 12 bits**
- ✓ **1 ou tous les canaux surveillés**
- ✓ **Interruption**

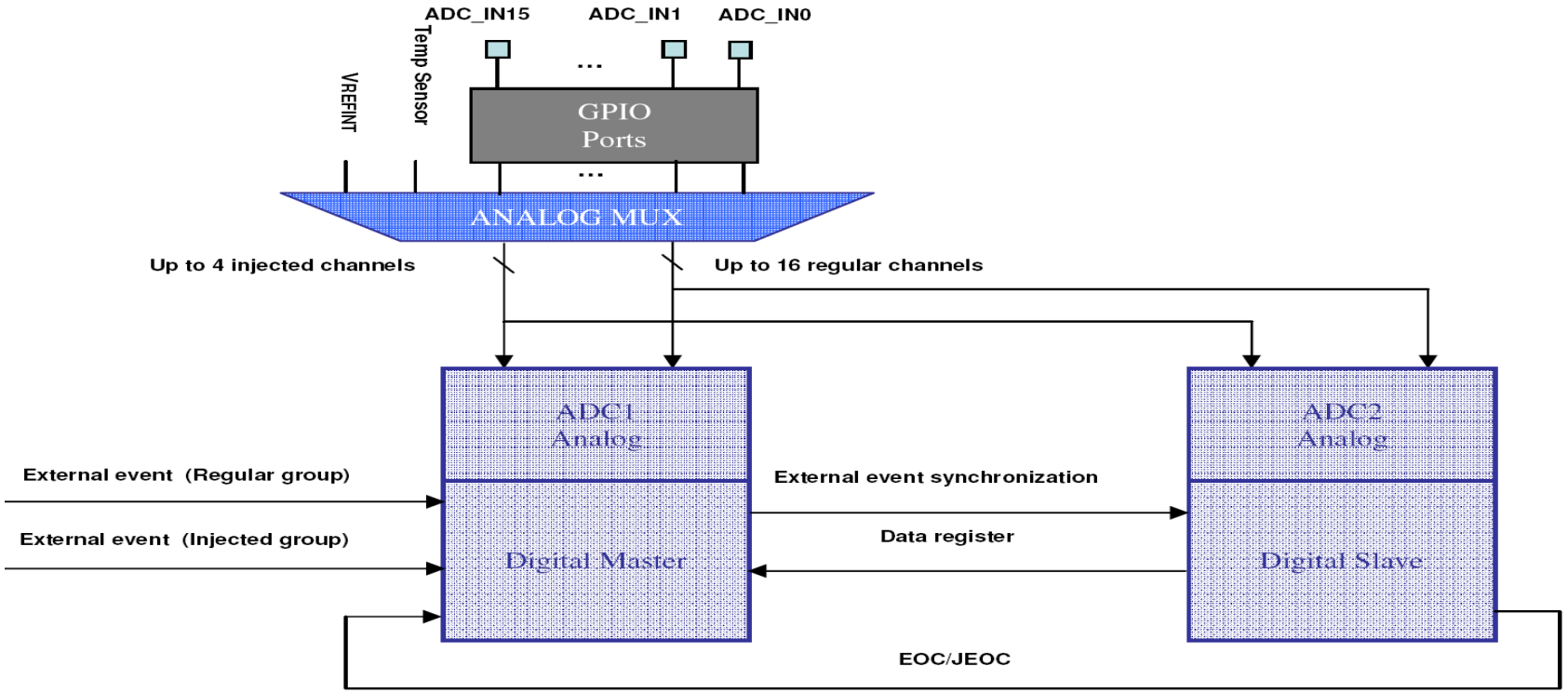




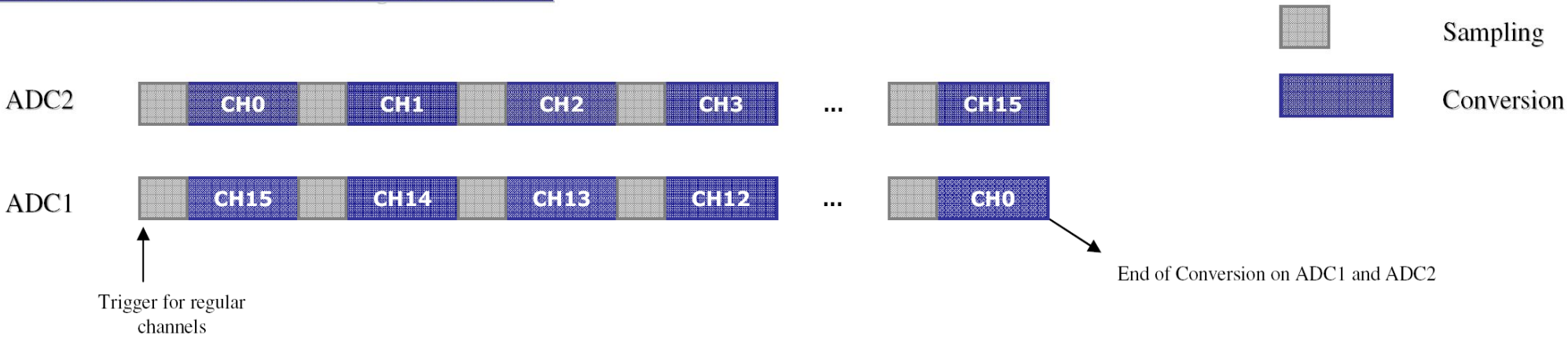
Example: - Conversion of Regular group
- DMA triggered by End of Conversion
- Results transferred to SRAM array by DMA
- DMA Destination address auto incremented
- EOC flag cleared by DMA access to ADCR1_DR

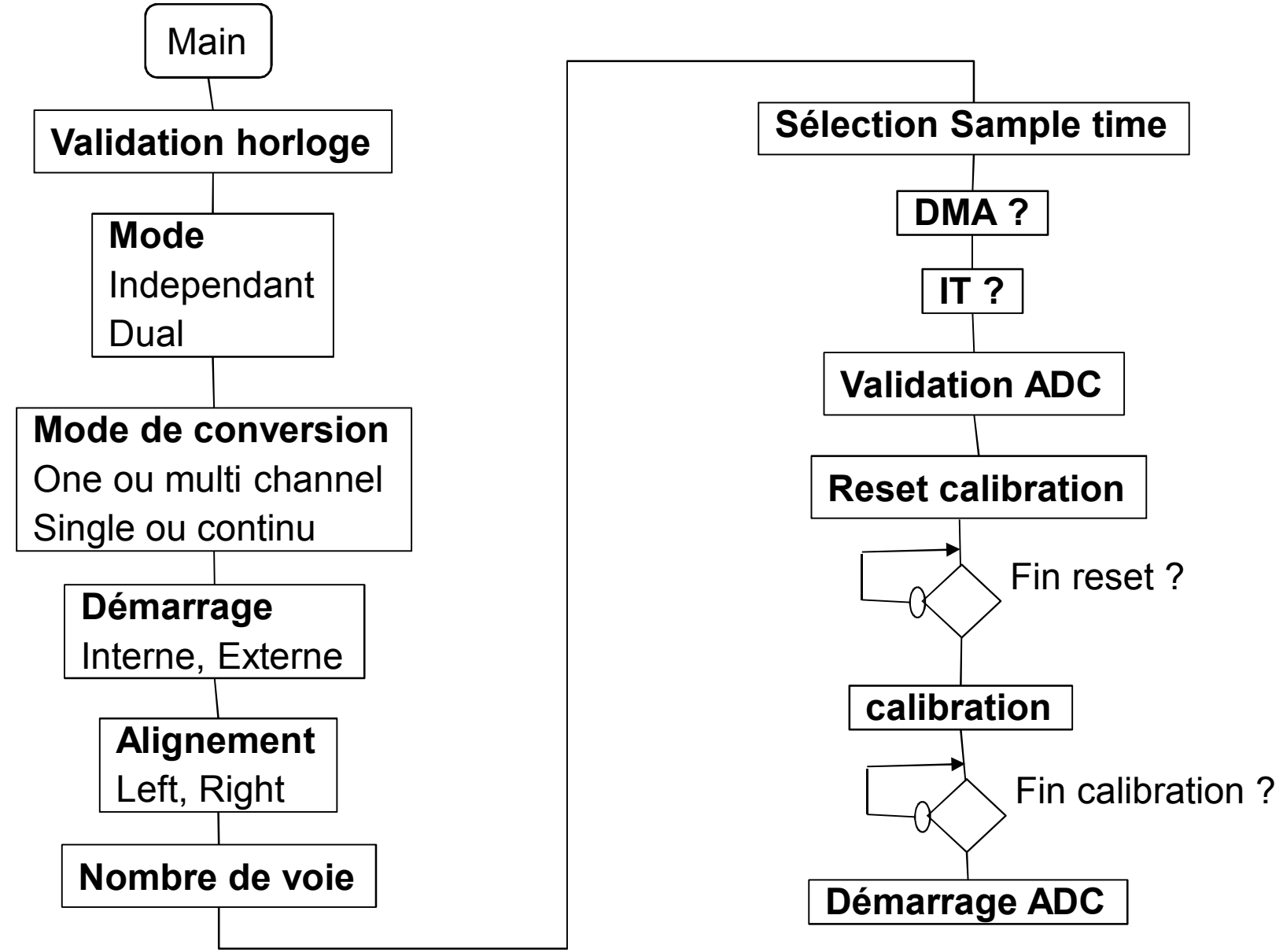


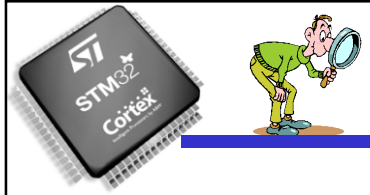
ADC DUAL MODE



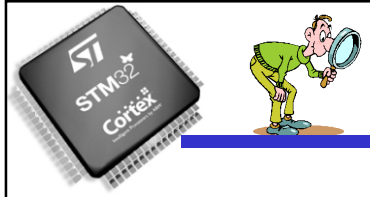
Regular simultaneous mode on 16 regular channels




Main
Validation horloge
Mode
Independant
Dual
Mode de conversion
One ou multi channel
Single ou continu
Démarrage
Interne, Externe
Alignement
Left, Right
Nombre de voie
Sélection Sample time
DMA ?
IT ?
Validation ADC
Reset calibration
Fin reset ?
calibration
Fin calibration ?
Démarrage ADC



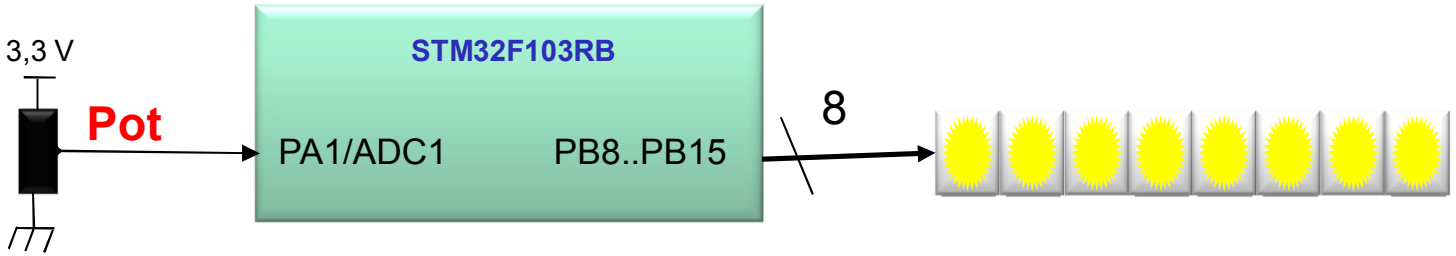

```
ADC_InitTypeDef     ADC_InitStructure;
ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_InitStructure.ADC_ScanConvMode = DISABLE;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None ;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Left;
ADC_InitStructure.ADC_NbrOfChannel = 1;
ADC_Init( ADC1, &ADC_InitStructure);
ADC_RegularChannelConfig( ADC1, ADC_Channel_1, 1, ADC_SampleTime_1Cycles5);
ADC_DMACmd(ADC1, ENABLE); // à chaque fin de conversion, demande de transfert DMA.
ADC_Cmd(ADC1, ENABLE );
ADC_ResetCalibration(ADC1); /* Enable ADC1 reset calibration register */
while(ADC_GetResetCalibrationStatus(ADC1)); /* Check the end of ADC1 reset calibration register */
ADC_StartCalibration(ADC1); /* Start ADC1 calibration */
 while(ADC_GetCalibrationStatus(ADC1)); /* Check the end of ADC1 calibration */
 ADC_SoftwareStartConvCmd( ADC1, ENABLE );
```



- Combien de voies analogiques possède le STM32 ?
- Quelle est la fréquence maximal de fonctionnement de l'ADC ?
- Sur quel bus est connecté l'ADC ?
- Quels sont les différents modes de conversion de l'ADC ?
- Pourquoi le temps d'acquisition est programmable ?
- Quel est l'intérêt du DMA sur l'ADC ?
- Quels sont les interruptions de l'ADC ?
- Combien de voies peuvent être surveillées par l'analog watchdog ?
- Intérêt de la calibration ?

* FONCTION: Convertir la tension issu du potentiomètre du kit.

* Afficher cette tension sur les 8 leds du kit.



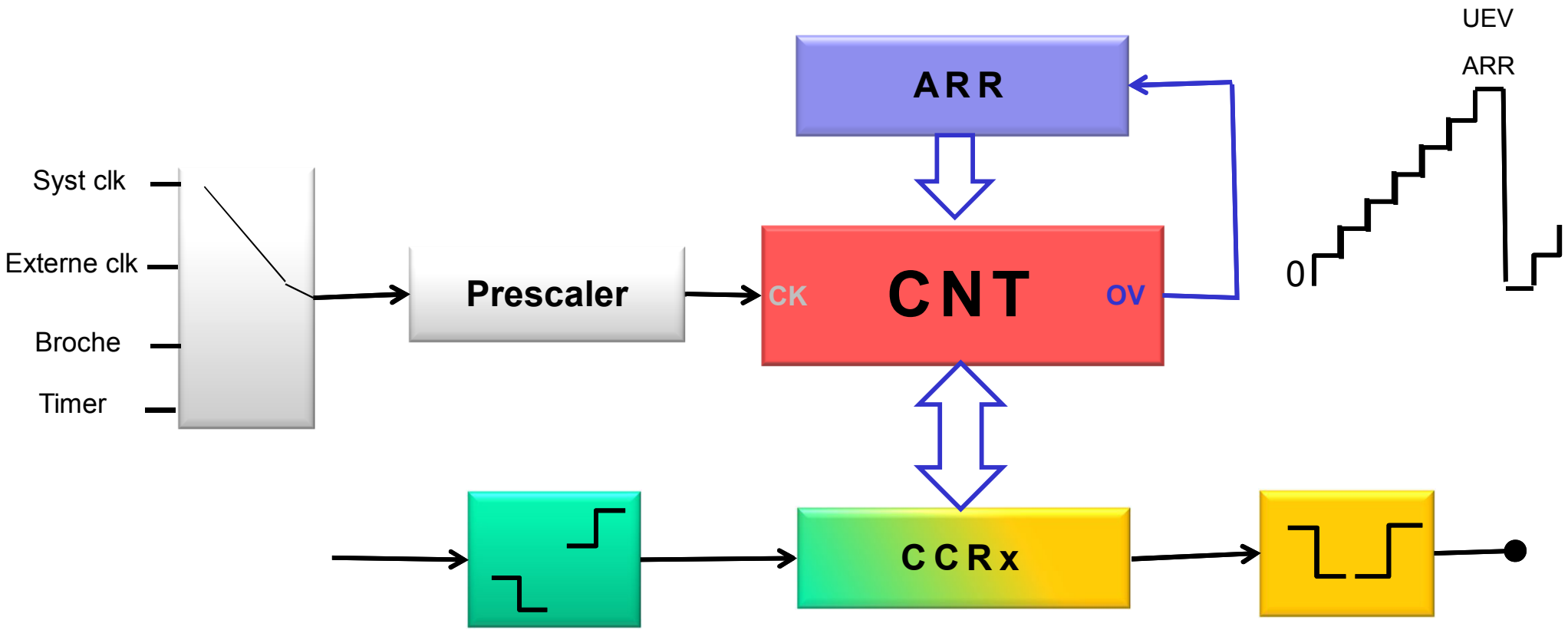
* FONCTION: Convertir la tension issu du potentiomètre du kit.

* Remplir un buffer de 256 échantillons via le dma.

* Faire la moyenne de ces 256 échantillons.

* Afficher cette moyenne sur les 8 leds du kit.

* Pour accélérer le traitement on utilisera un buffer circulaire de 512 échantillons.

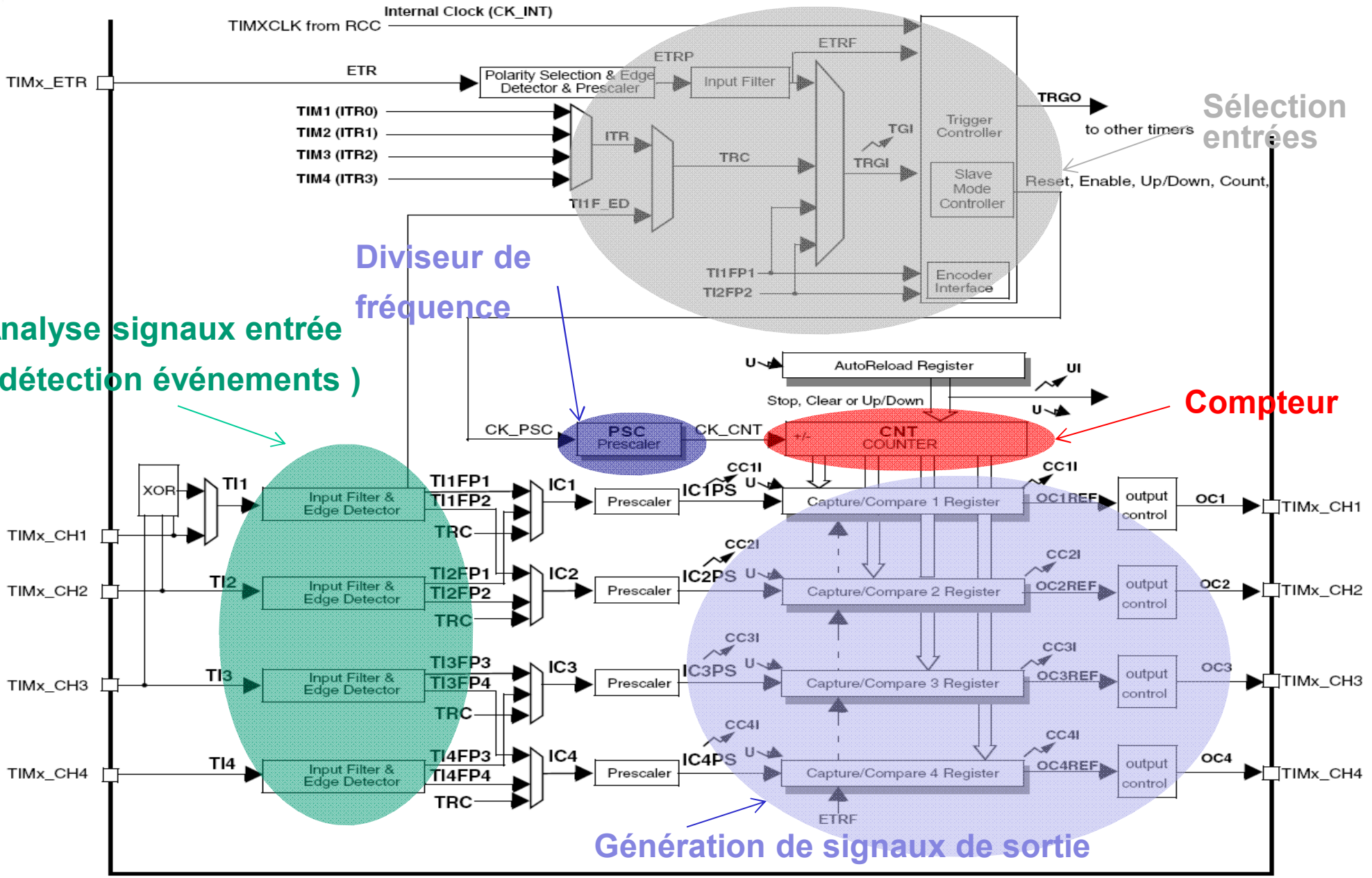


- **Capture**

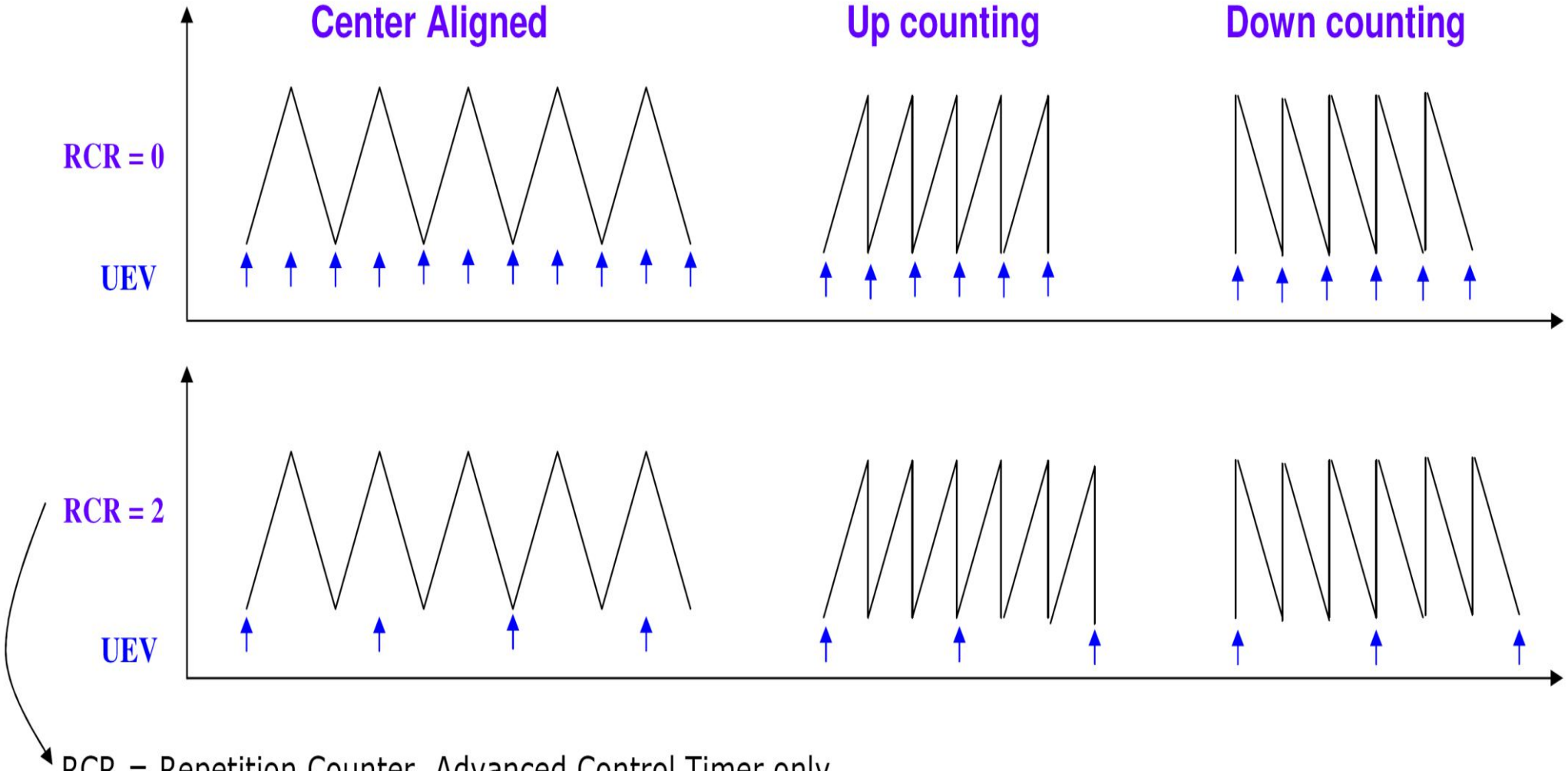
  Mesure signaux

- **Comparaison**

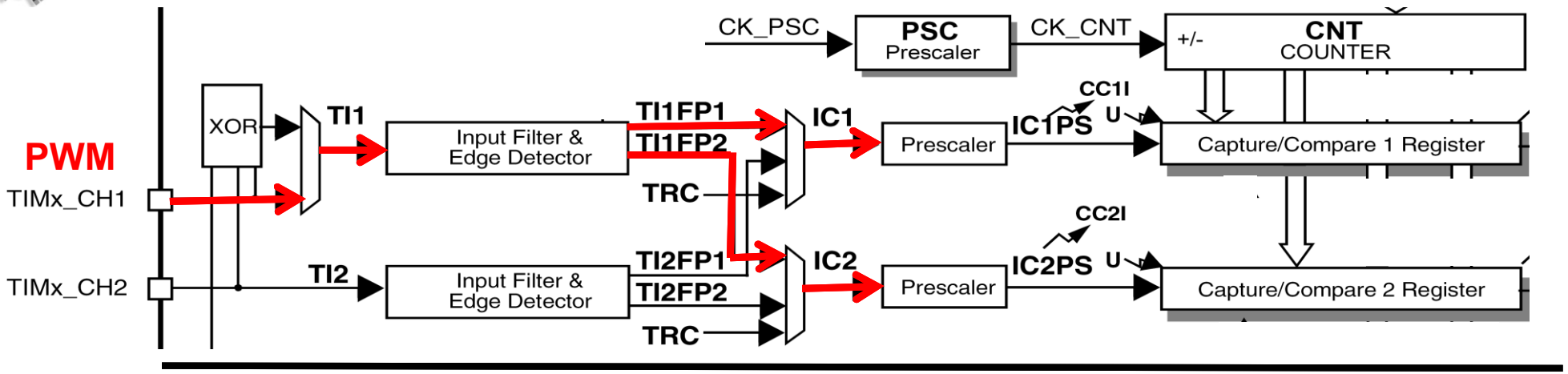
  Génération signaux

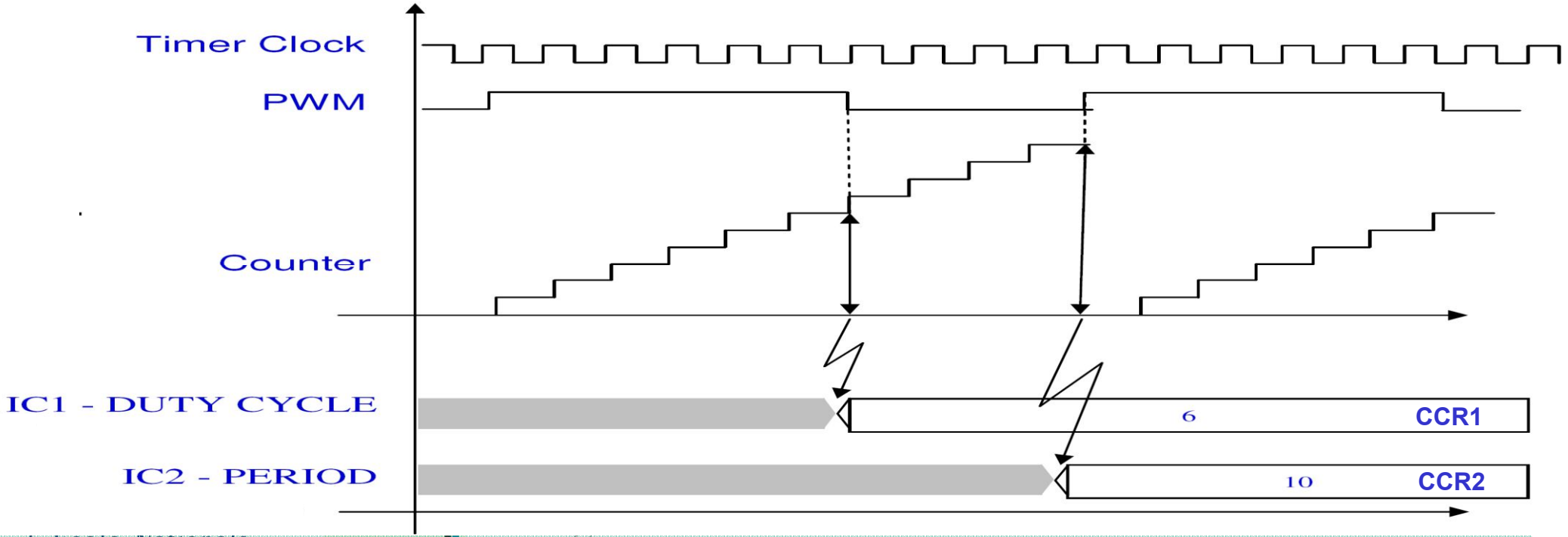

Internal Clock (CK_INT)
TIMXCLK from RCC
TIMx_ETR
ETR
Polarity Selection & Edge Detector & Prescaler
ETRP
Input Filter
ETRF
TIM1 (ITR0)
TIM2 (ITR1)
TIM3 (ITR2)
TIM4 (ITR3)
ITR
TRC
TI1F_ED
TGI
TRGI
Trigger Controller
TRGO
to other timers
Slave Mode Controller
Reset, Enable, Up/Down, Count,
TI1FP1
TI2FP2
Encoder Interface
Sélection entrées
Diviseur de fréquence
Analyse signaux entrée (détection événements)
Compteur
AutoReload Register
UI
Stop, Clear or Up/Down
CK_PSC
PSC Prescaler
CK_CNT
+/-
CNT COUNTER
XOR
TI1
TI2
TI3
TI4
Input Filter & Edge Detector
TI1FP1
TI1FP2
TI2FP1
TI2FP2
TI3FP3
TI3FP4
TI4FP3
TI4FP4
IC1
IC2
IC3
IC4
Prescaler
IC1PS
IC2PS
IC3PS
IC4PS
CC1I
CC2I
CC3I
CC4I
Capture/Compare 1 Register
Capture/Compare 2 Register
Capture/Compare 3 Register
Capture/Compare 4 Register
OC1REF
OC2REF
OC3REF
OC4REF
output control
OC1
OC2
OC3
OC4
TIMx_CH1
TIMx_CH2
TIMx_CH3
TIMx_CH4
Génération de signaux de sortie (capture du timer)

MODE DE COMPTAGE

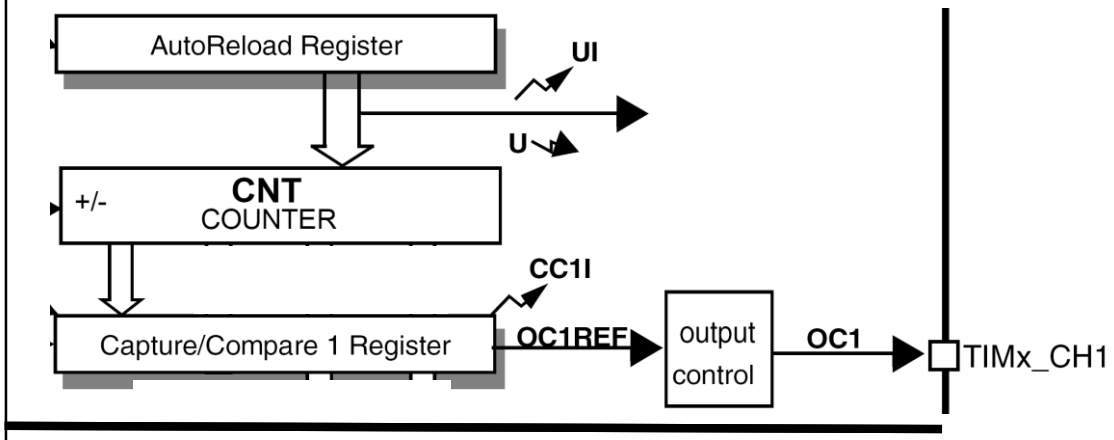


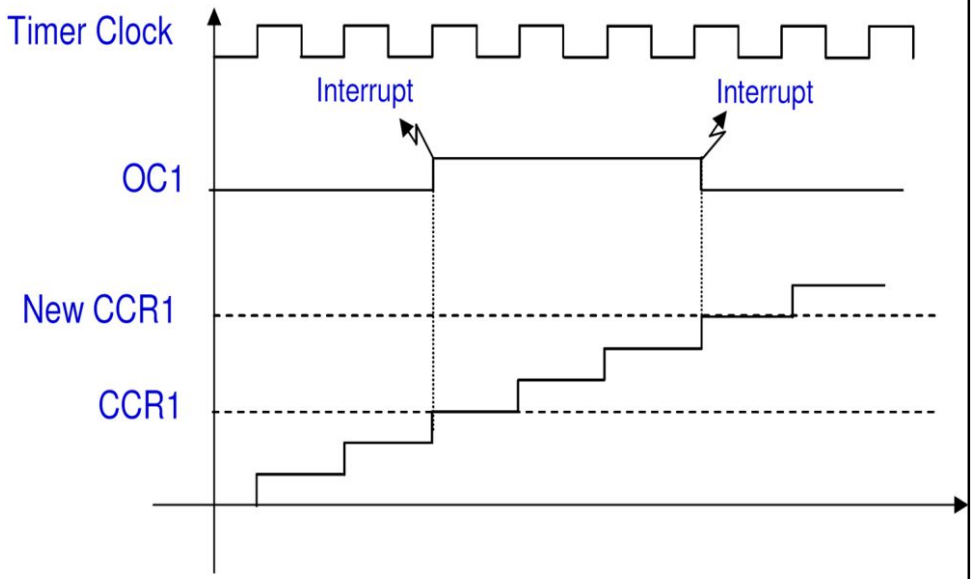
INPUT CAPTURE MODE



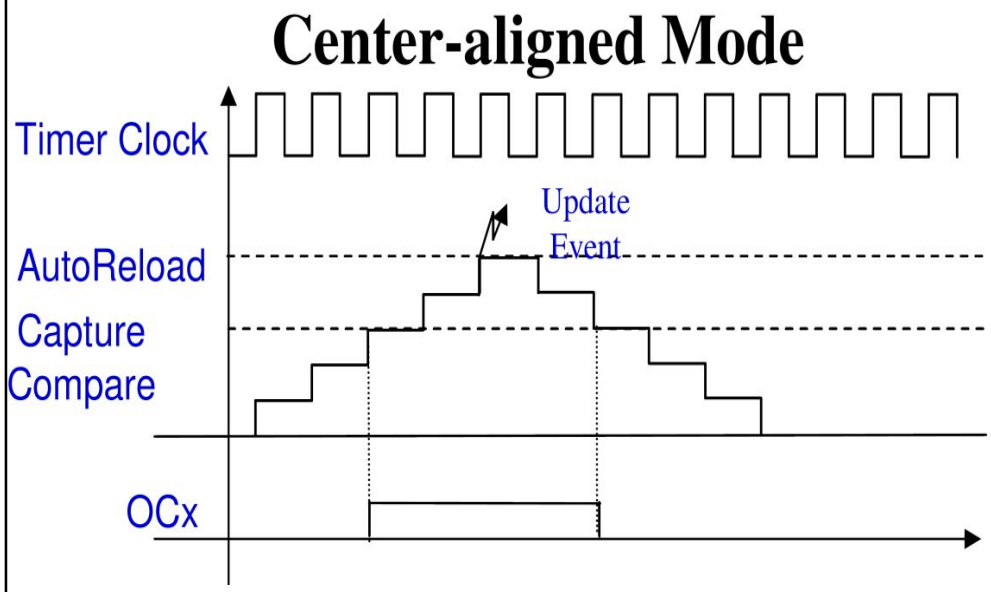


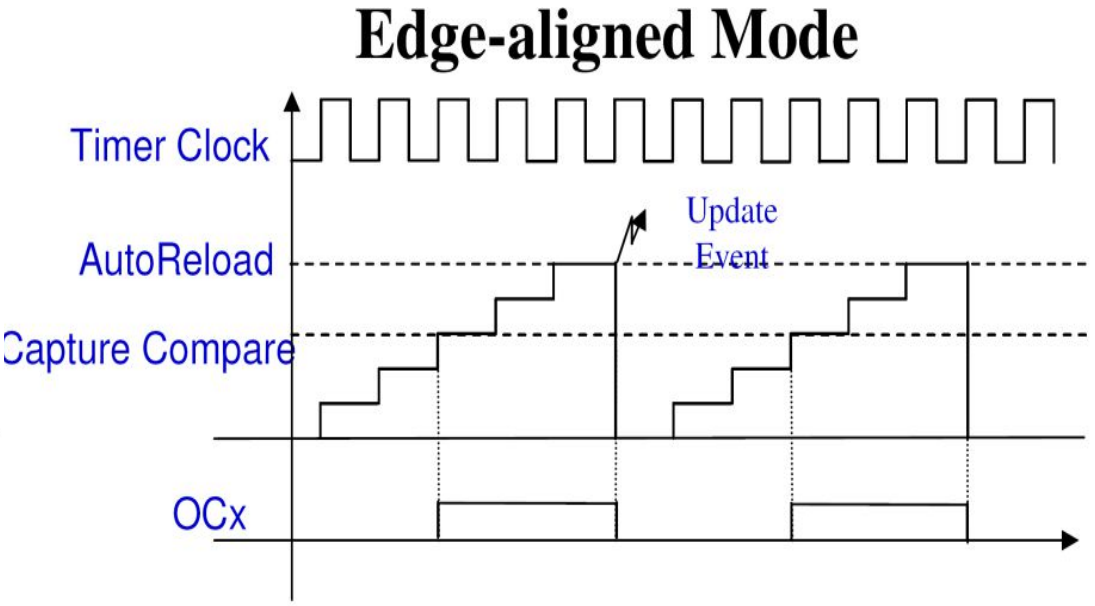
OUTPUT COMPARE MODE



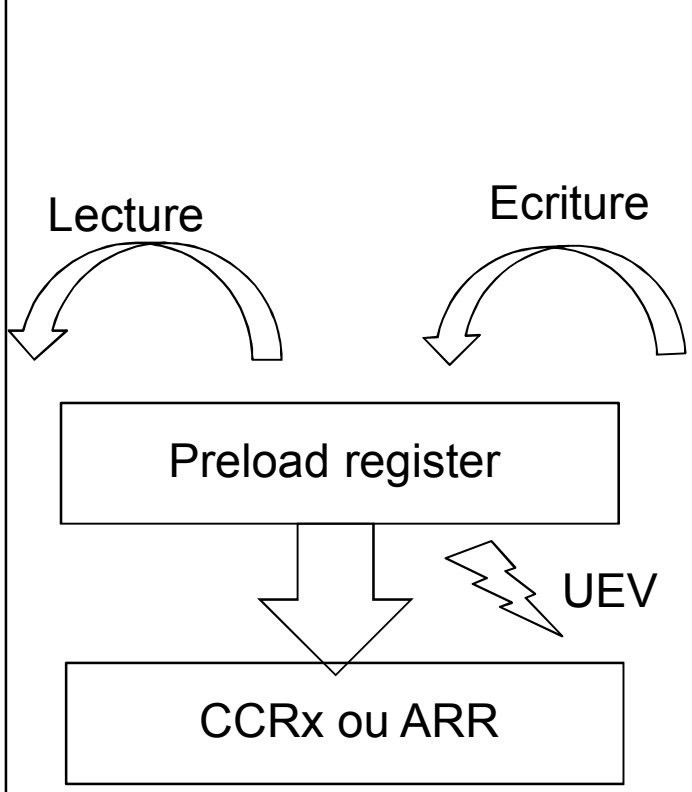


- **PWM MODE**



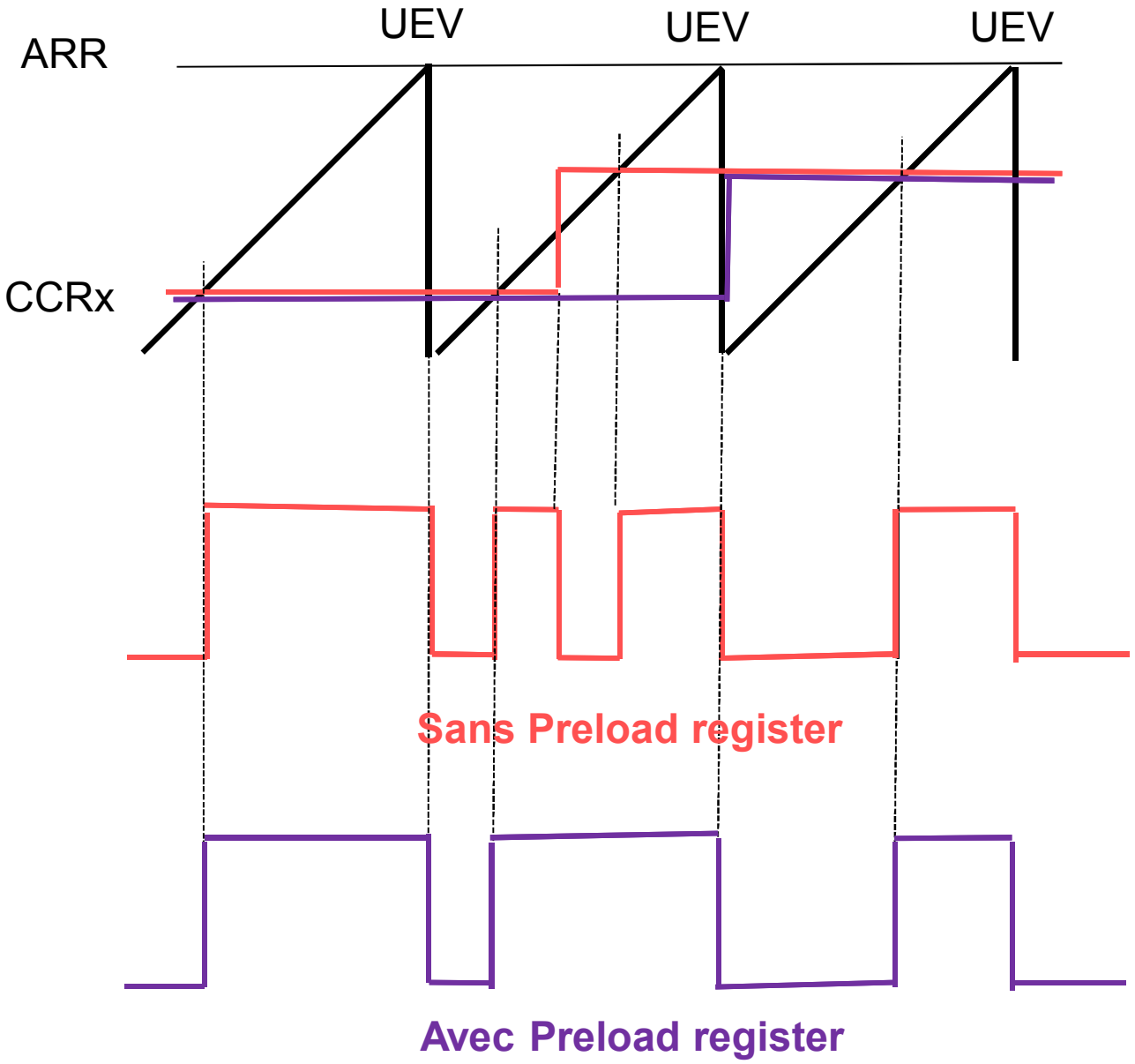


PRELOAD REGISTER

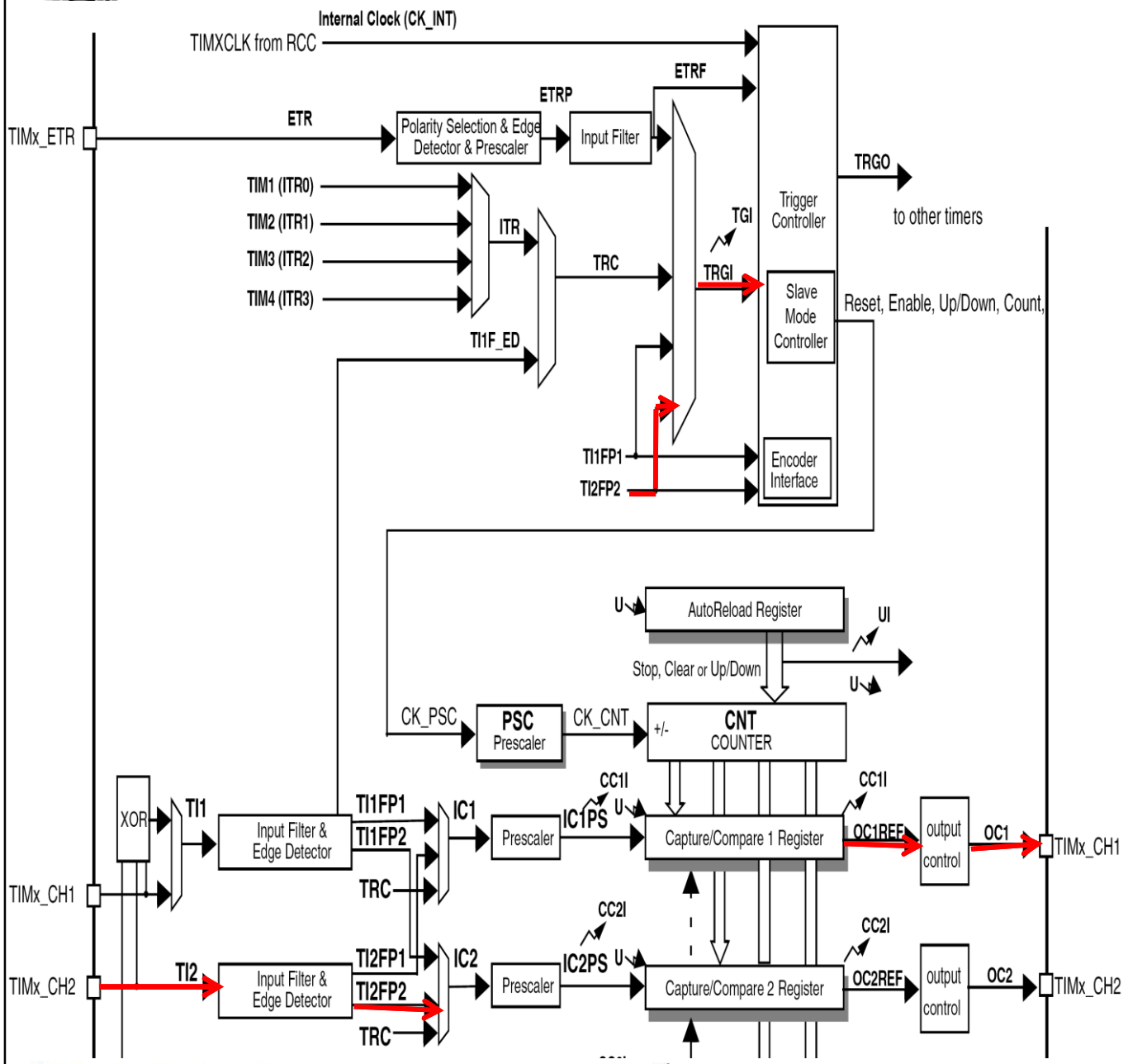


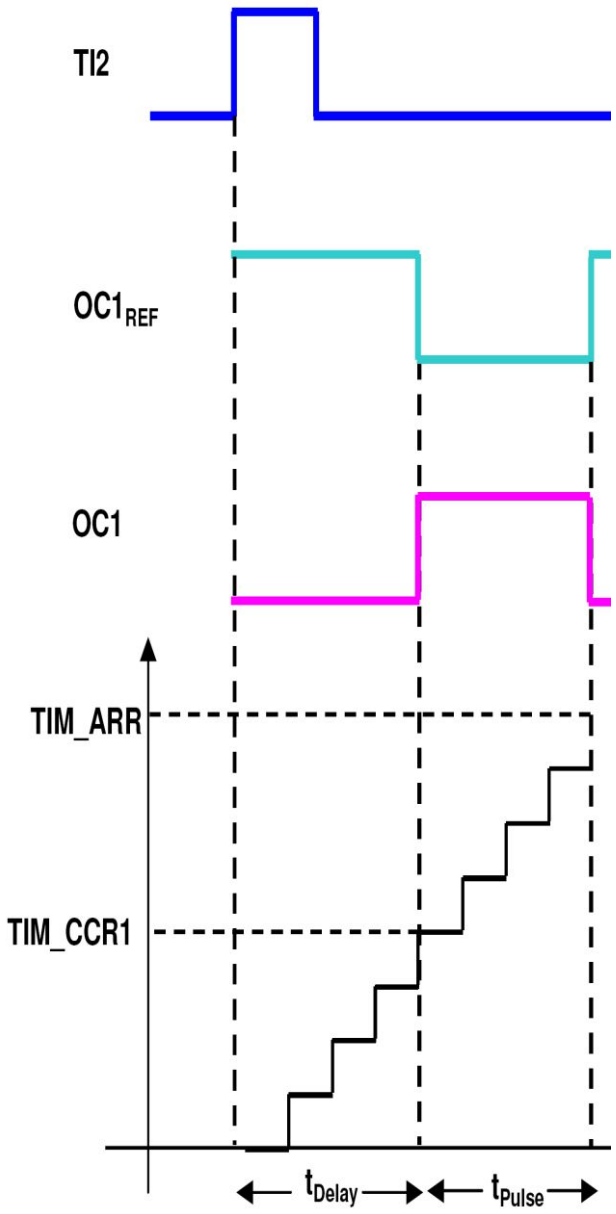
Chargement direct ou

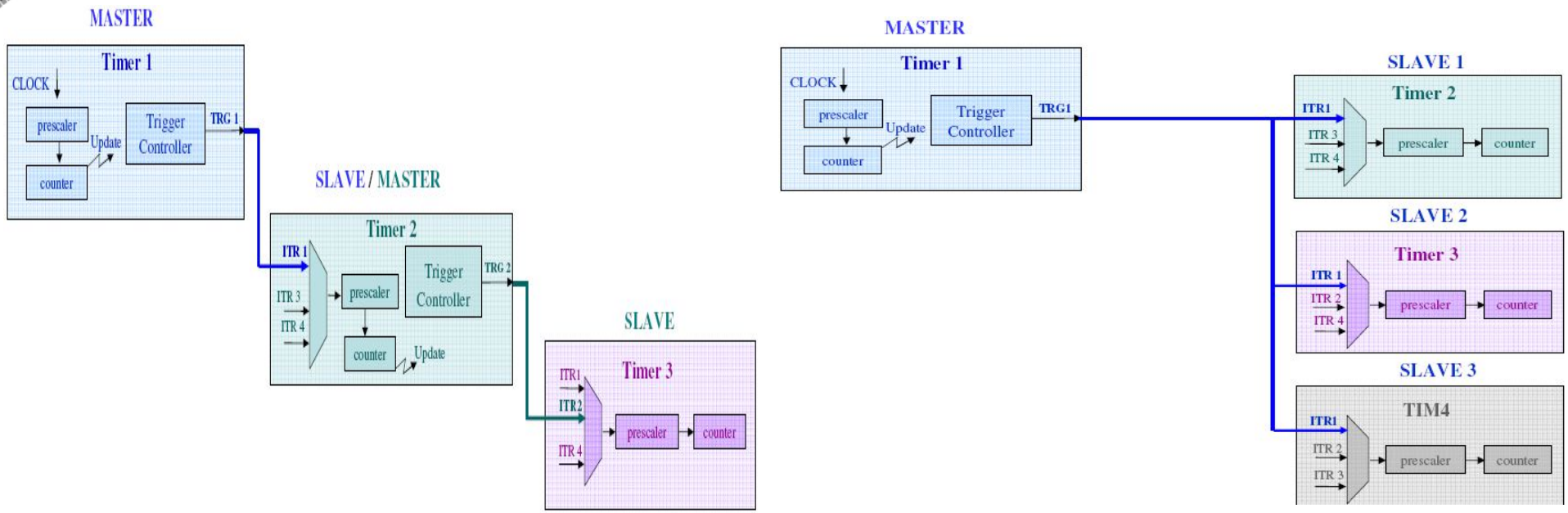
à chaque UEV

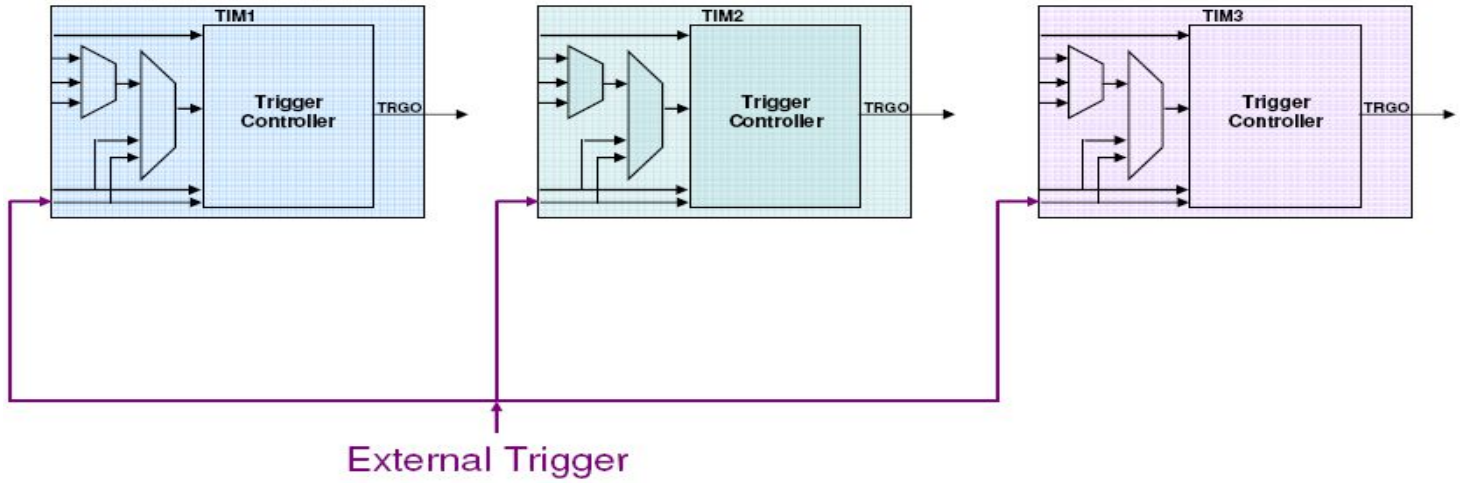


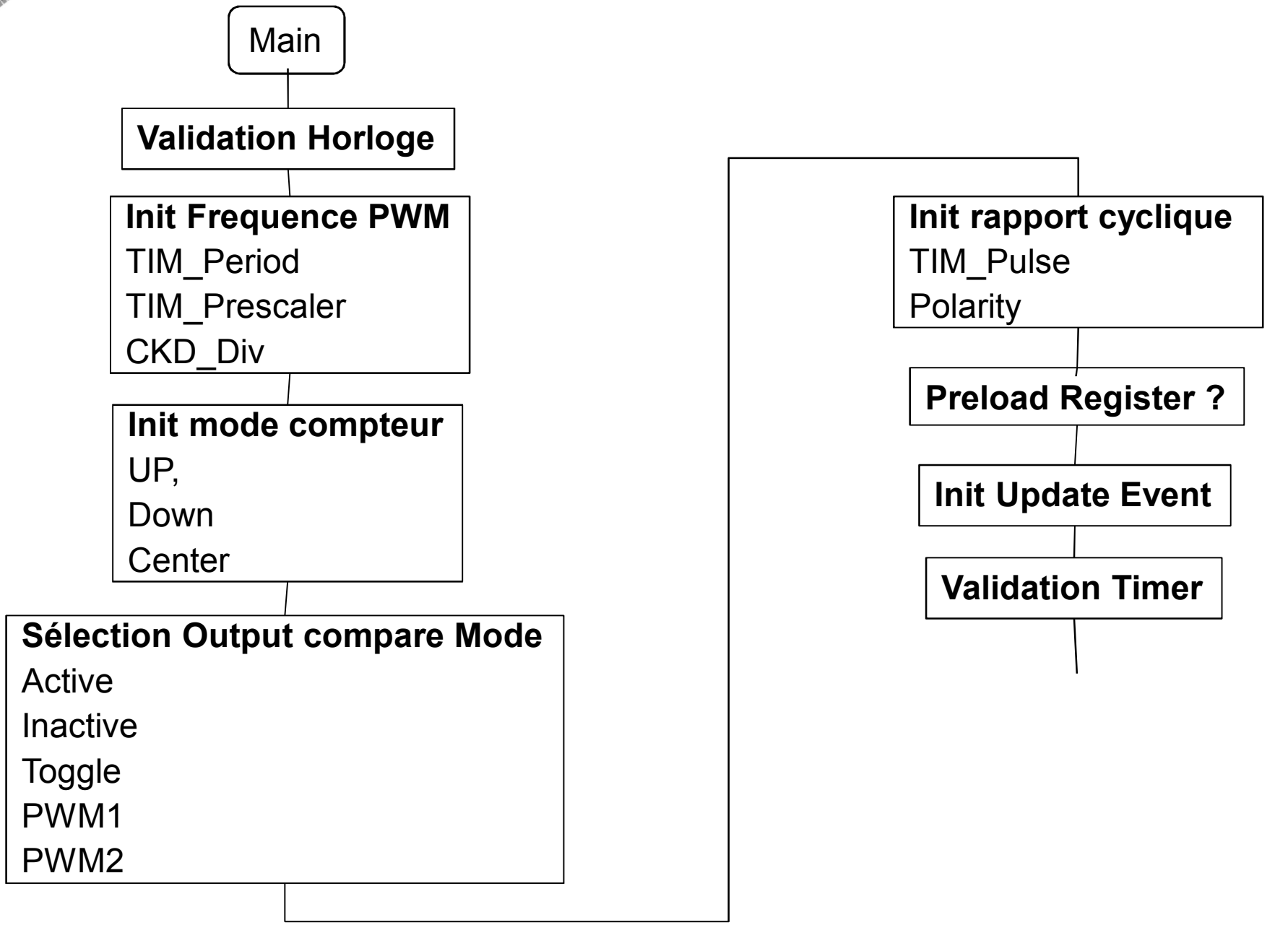
ONE PULSE MODE






MASTER
Timer 1
CLOCK
prescaler
Update
Trigger Controller
TRG 1
counter
SLAVE / MASTER
Timer 2
ITR 1
ITR 3
ITR 4
prescaler
Trigger Controller
TRG 2
counter
Update
SLAVE
Timer 3
ITR1
ITR2
ITR 4
prescaler
counter
MASTER
Timer 1
CLOCK
prescaler
Update
Trigger Controller
TRG1
counter
SLAVE 1
Timer 2
ITR1
ITR 3
ITR 4
prescaler
counter
SLAVE 2
Timer 3
ITR 1
ITR 2
ITR 4
prescaler
counter
SLAVE 3
TIM4
ITR1
ITR 2
ITR 3
prescaler
counter


TIM1
Trigger Controller
TRGO
TIM2
Trigger Controller
TRGO
TIM3
Trigger Controller
TRGO
External Trigger

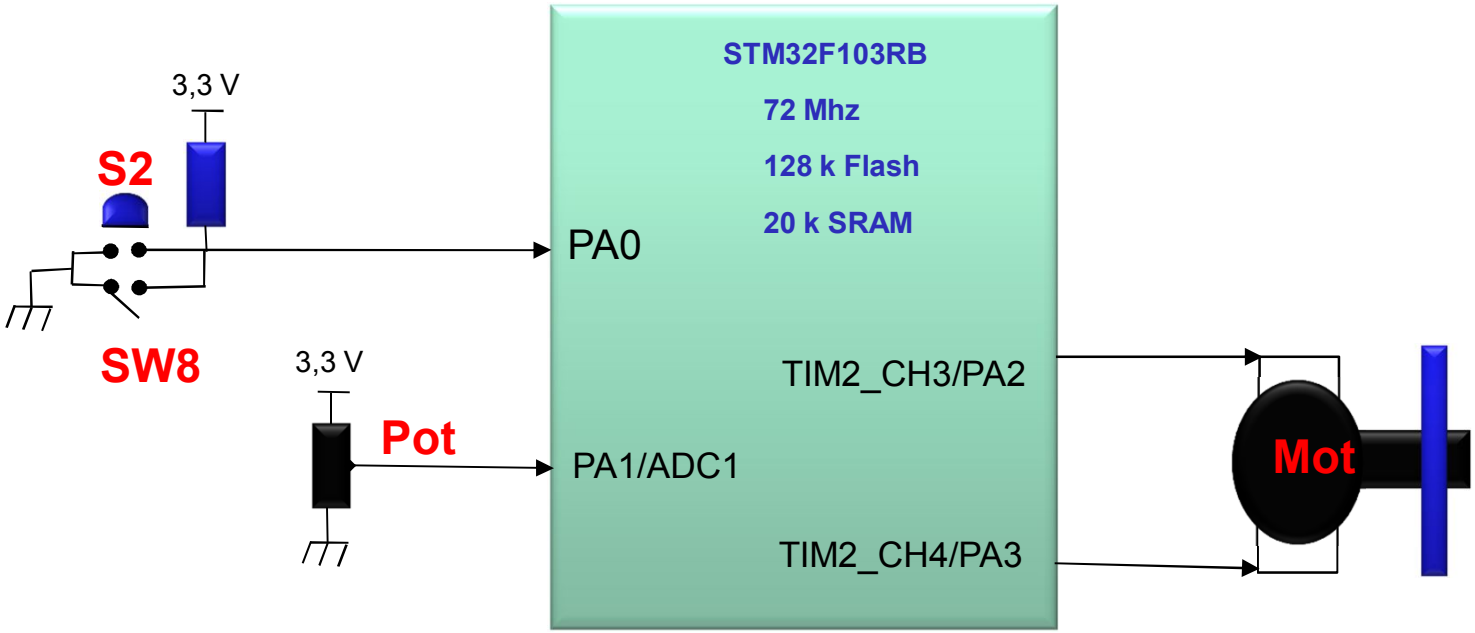

Main
Validation Horloge
Init Frequence PWM
TIM_Period
TIM_Prescaler
CKD_Div
Init mode compteur
UP,
Down
Center
Sélection Output compare Mode
Active
Inactive
Toggle
PWM1
PWM2
Init rapport cyclique
TIM_Pulse
Polarity
Preload Register ?
Init Update Event
Validation Timer

```
void Timer_init() {

  TIM_TimeBaseInitTypeDef      TIM_TimeBaseStructure;
  TIM_OCInitTypeDef            TIM_OCInitStructure;

  u16 TIM_period = 999;
  u16 TIM_Prescaler = 72;

/* Time base configuration */
  TIM_InternalClockConfig(TIM2);
 TIM_TimeBaseStructure.TIM_Period = TIM_period;
 TIM_TimeBaseStructure.TIM_Prescaler = TIM_Prescaler;     /* Fpwm =  ????? */
 TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
 TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
```

```
/* PWM1 Mode configuration: Channel3 */

TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_Pulse = 0;                    / * Rapport cyclique  ?????? */
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC3Init(TIM2, &TIM_OCInitStructure);

  TIM_OC3PreloadConfig(TIM2, TIM_OCPreload_Enable);
  TIM_ARRPreloadConfig(TIM2, ENABLE);
  TIM_UpdateDisableConfig(TIM2, DISABLE);
  TIM_UpdateRequestConfig ( TIM2, TIM_UpdateSource_Regular);
  TIM_Cmd ( TIM2, ENABLE );
}
```

- Combien de timer possède le STM32 ?
- Sur quel bus sont connectés les timers ?
- Combien de voies de capture/ comparaison par timer ?
- Rôle de l'auto-reload register ?
- Quel est le principe de création d'un PWM ?
- Intérêt de la synchronisation entre timers ?
- Qu'appelle-t-on un Preload Register dans les Timers?
- Quand l'Update Event est généré dans les Timers?
- Pourquoi les Timers du STM32 sont particulièrement adaptés au contrôle moteur ?

```
*-----------------------------------------------------------------------------------------------------------
* FONCTION :  Commande PWM moteur 1 Khz, 10 % via le bouton poussoir S2
*                    ON: Marche
*                    OFF: Arrêt
*
*-----------------------------------------------------------------------------------------------------------
```



* FONCTION: Commande PWM moteur 1 Khz via le potentiomètre

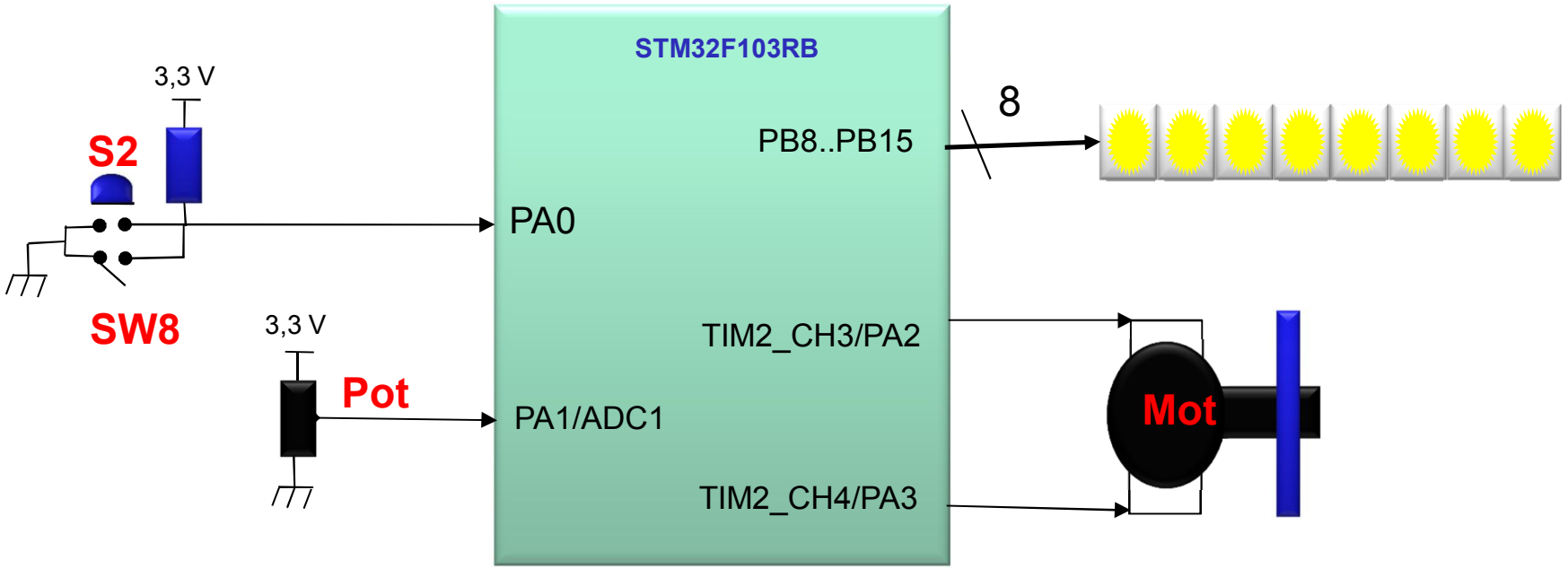
* Utiliser une voie DMA pour faire le transfert entre l'ADC et le Timer en mode

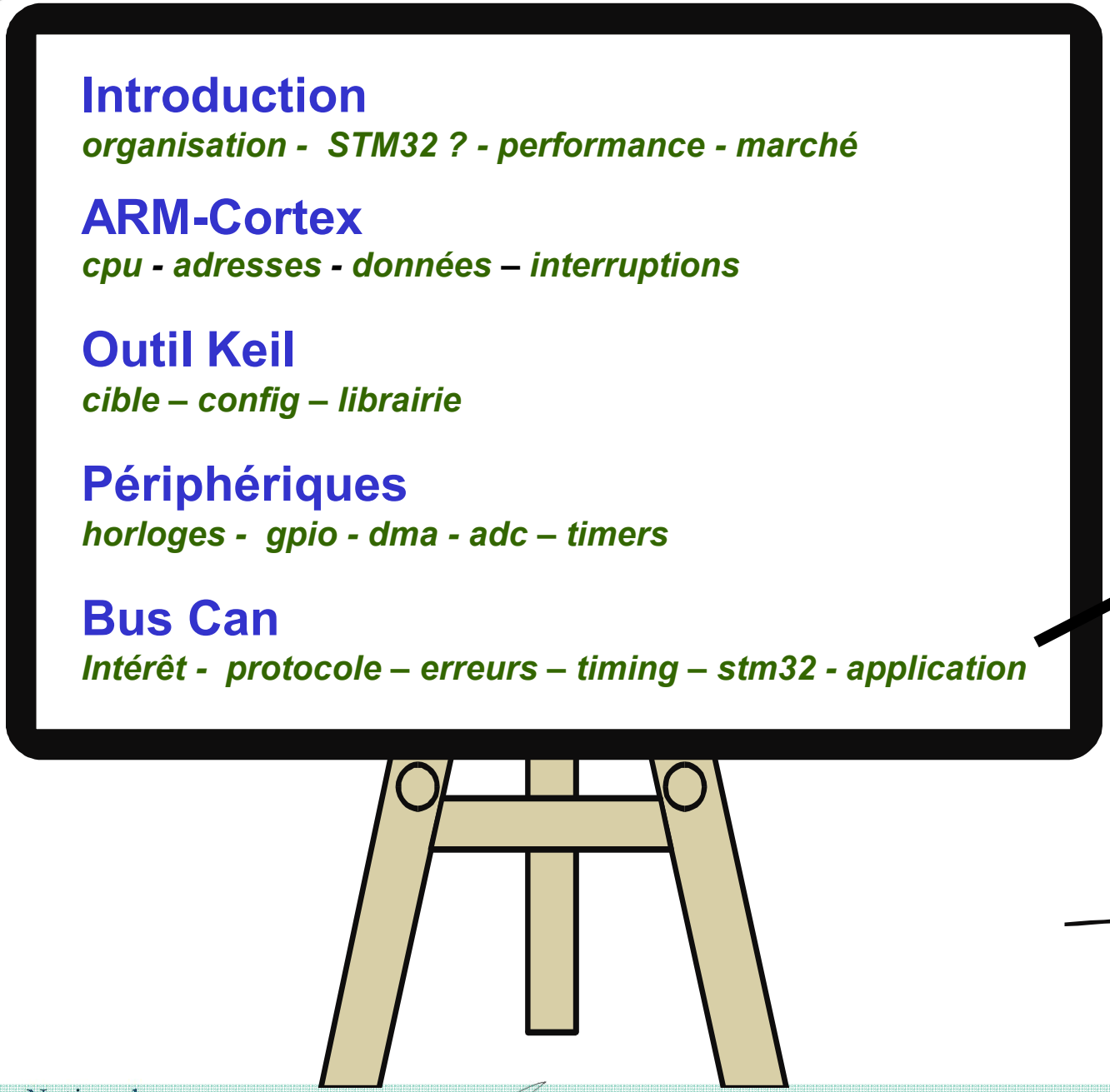
* PWM.

* Utiliser l'Analog Watchdog pour détecter les survitesses, Faire clignoter

* les 8 leds du kits pour signaler ce cas.

* En fonctionnement normal on utilisera les 8 leds pour afficher la commande.





Introduction

organisation - STM32 ? - performance - marché

ARM-Cortex

cpu - adresses - données – interruptions

Outil Keil

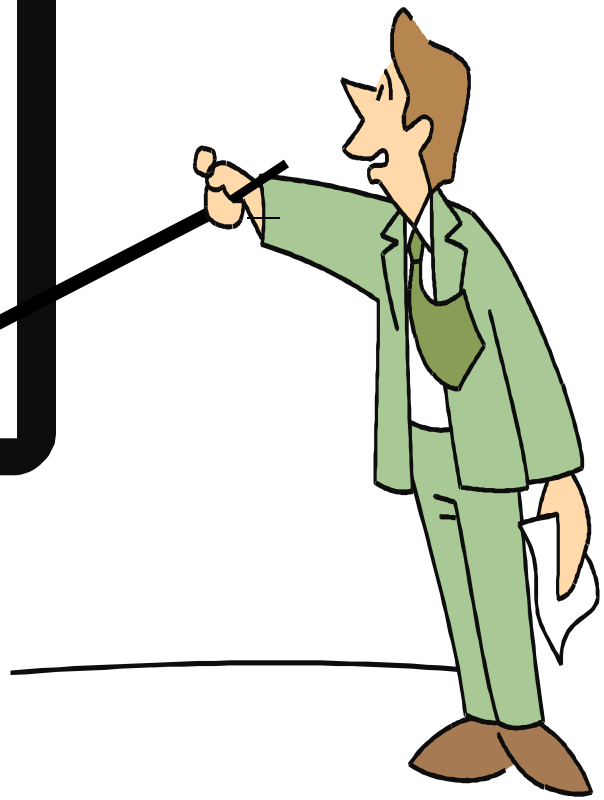
cible – config – librairie

Périphériques

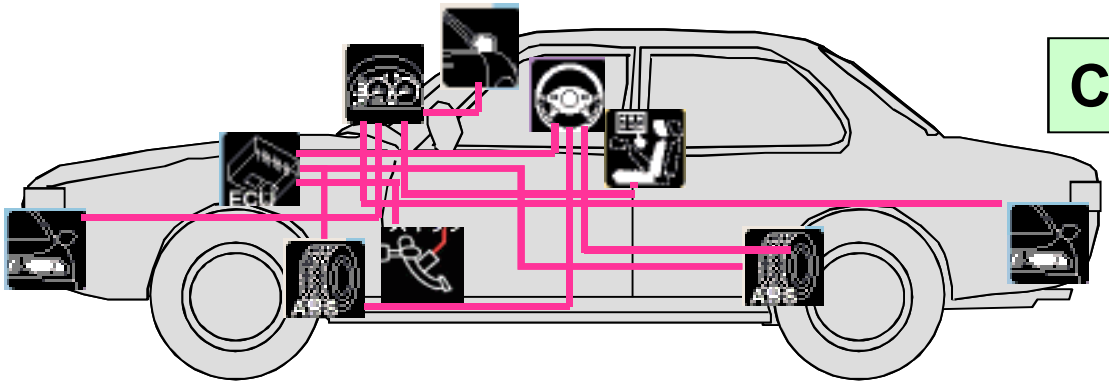
horloges - gpio - dma - adc – timers

Bus Can

Intérêt - protocole – erreurs – timing – stm32 - application

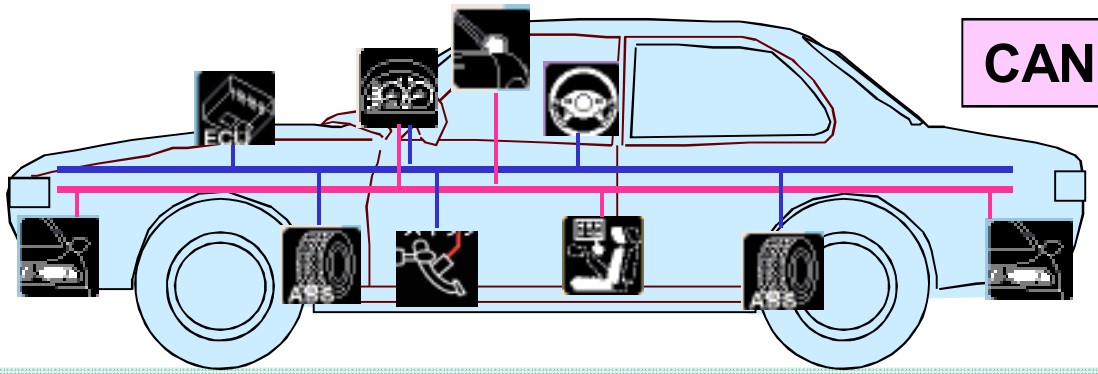


- ✓ BOSCH 1983 – normalisation ISO 1994
- ✓ Multiplexage des interconnections
- ✓ Haut niveau de sécurité
- ✓ Temps réel

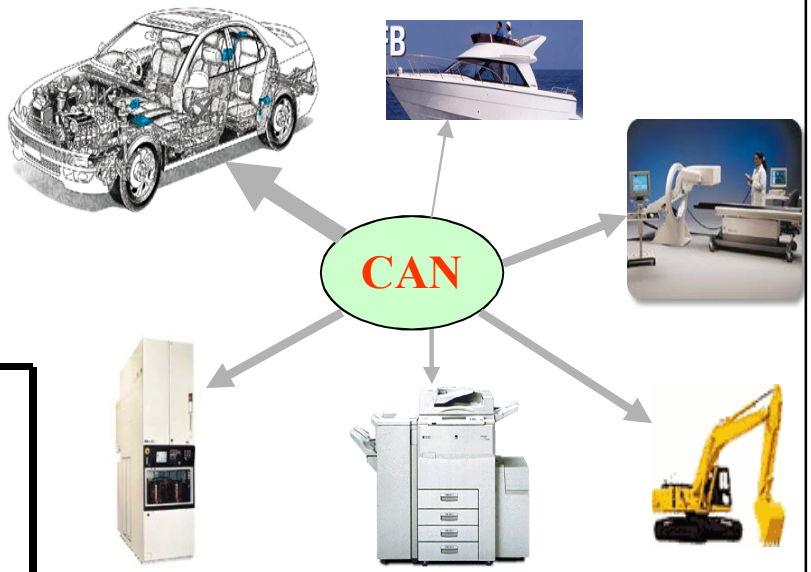


Câblage traditionnel

Simple & Léger



CAN



Débit

(kbits)

10 000

1 000

100

10

1

Ethernet

Réseau d 'ordinateurs

CAN

Automobile
application motorisation

I²C

D²B

Automobile
application confort

Automobile
application éclairage

Application
bus de terrain industriel

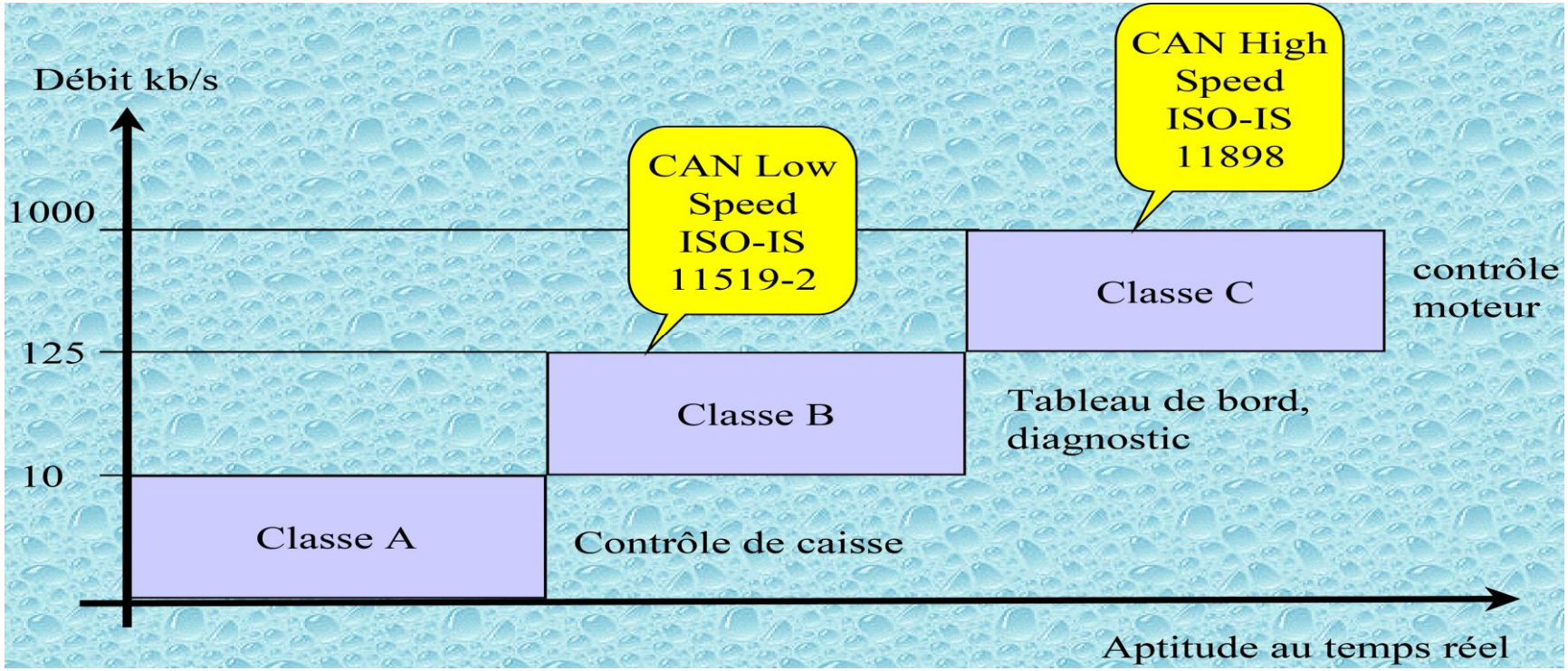
1

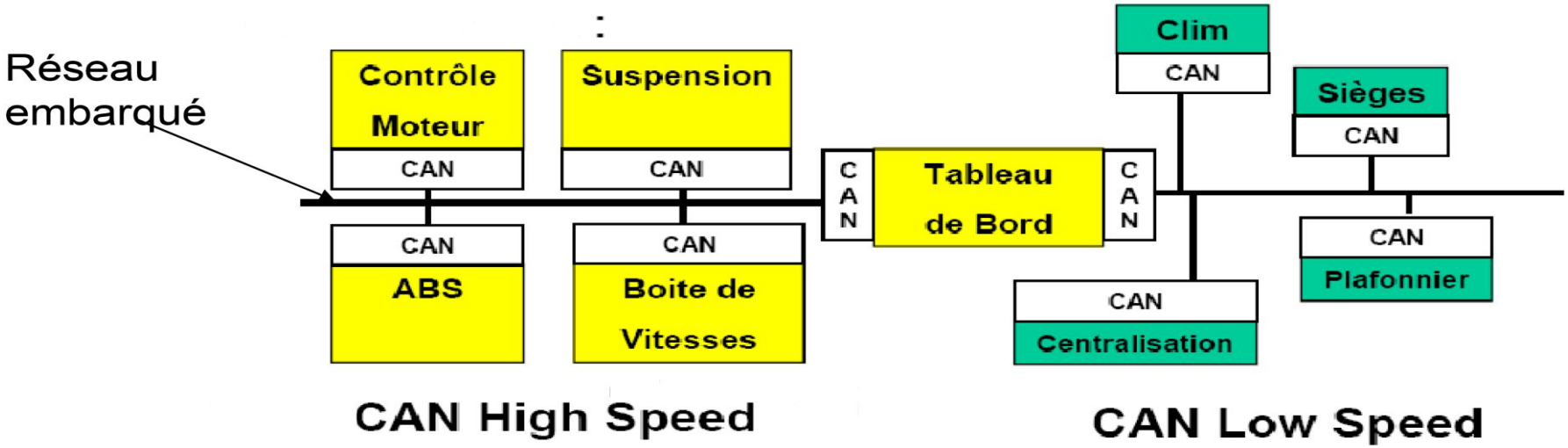
10

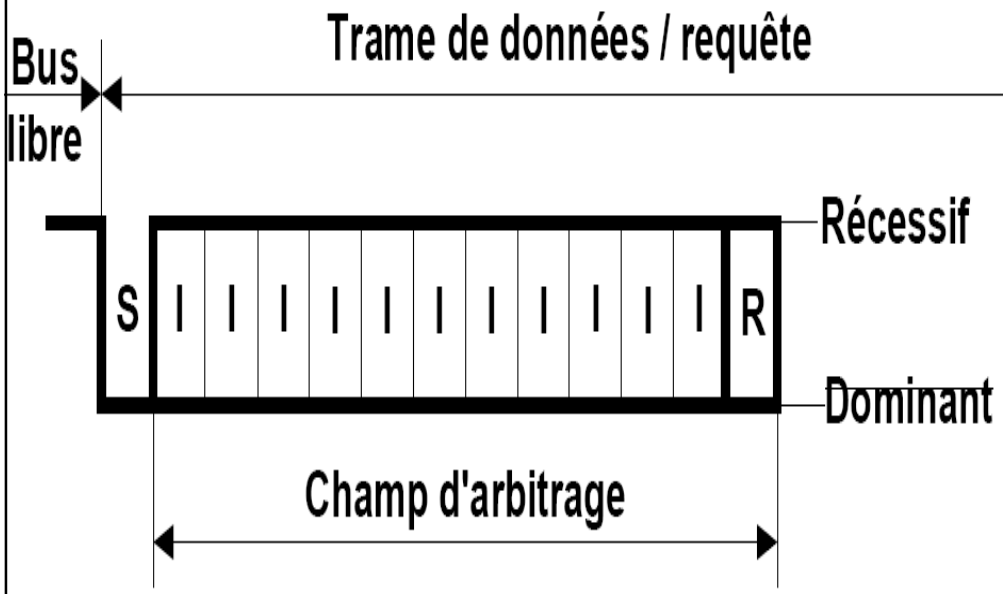
100

1 000

10 000 Distance (m)


Débit kb/s
1000
125
10
CAN High Speed ISO-IS 11898
CAN Low Speed ISO-IS 11519-2
Classe C
contrôle moteur
Classe B
Tableau de bord, diagnostic
Classe A
Contrôle de caisse
Aptitude au temps réel


Réseau embarqué
Contrôle Moteur
CAN
Suspension
CAN
CAN
Tableau de Bord
CAN
Clim
CAN
Sièges
CAN
CAN
ABS
CAN
Boite de Vitesses
CAN
Centralisation
CAN
Plafonnier
CAN High Speed
CAN Low Speed

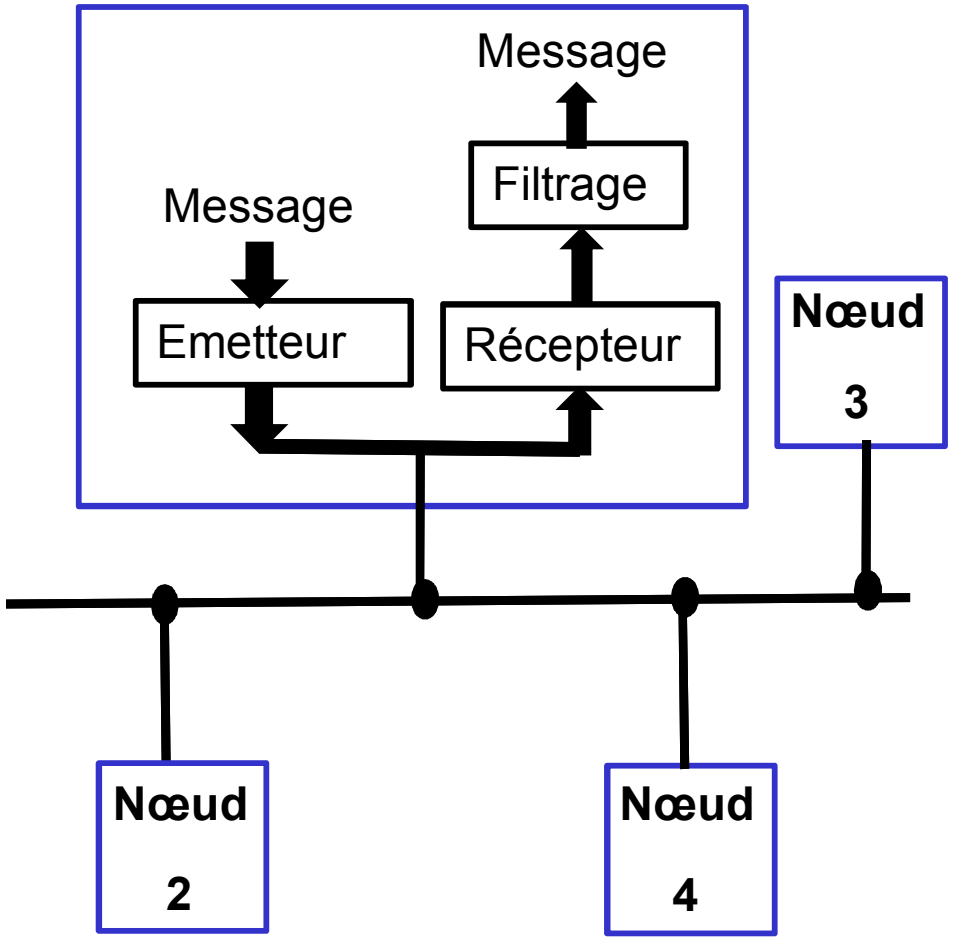


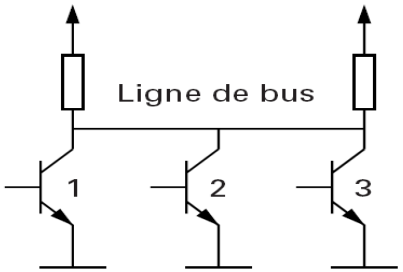
I : 11 bits d'identificateur

R : Bit RTR

- récessif : trame de requête
- dominant : trame de données

S : bit de départ de trame





Champ d'arbitrage

S O F | Identificateur 10 9 8 7 6 5 4 3 2 1 0 | R T R | Champ de commande | Champ de données 0 à 8 octets

Station 1

Station 2

Station 3

La station 2 perd l'arbitrage

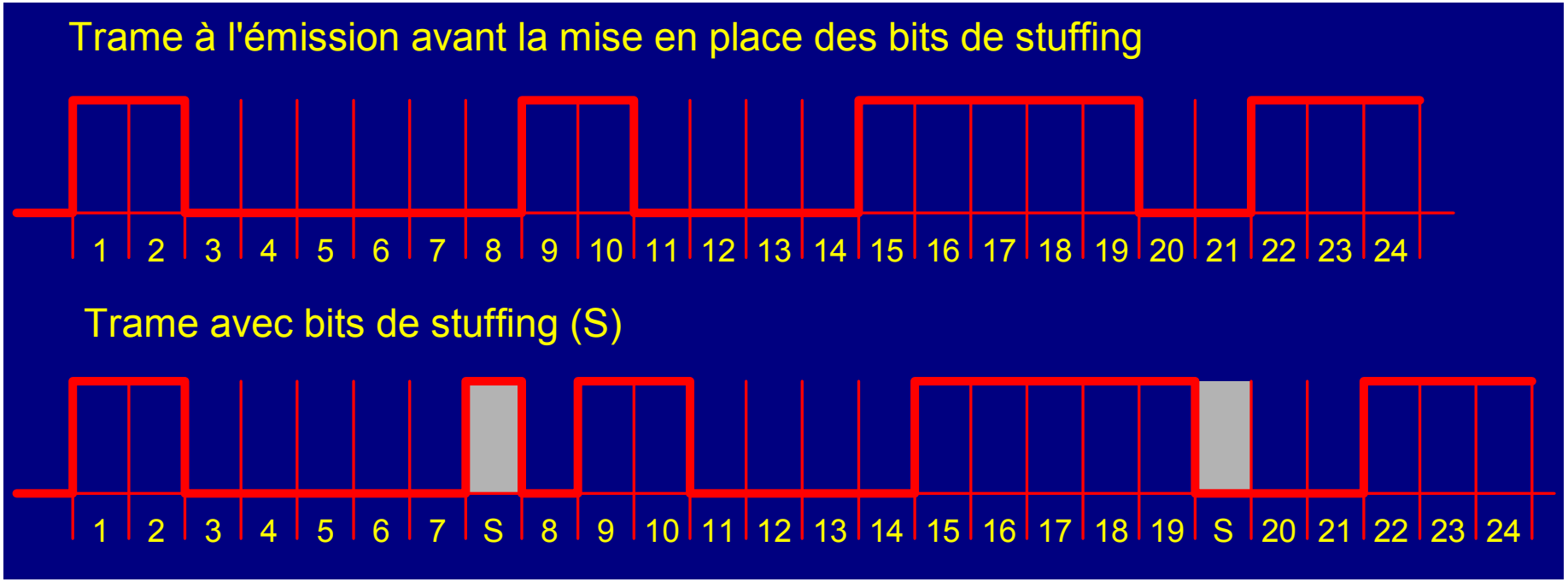
La station 1 perd l'arbitrage

Signal sur le bus

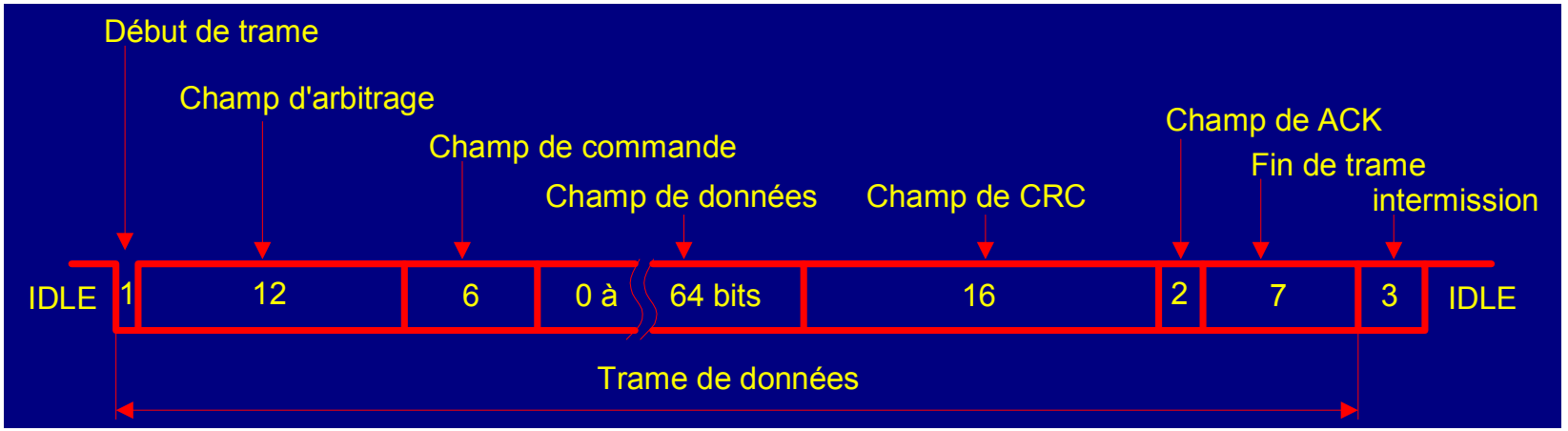
La station 3 a gagne le bus

BIT STUFFING

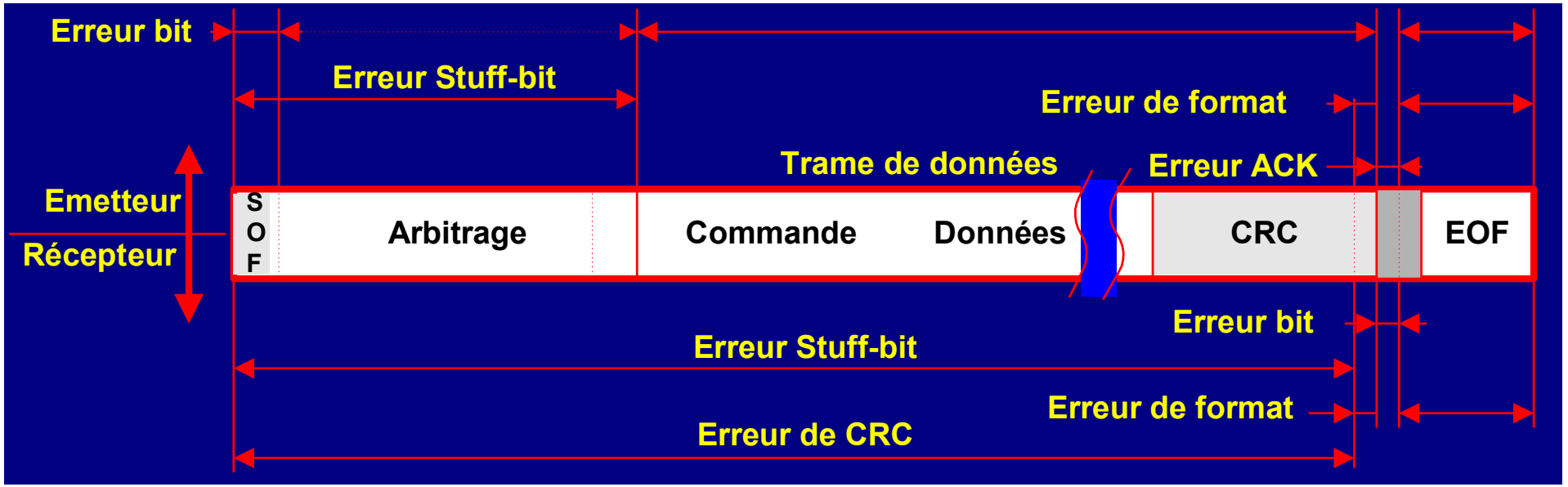
- ✓ But : détecter les anomalies de réseau
- ✓ Principe: ajout d'un bit complémentaire tous les 5 bits identiques



- **Données** *« Bonjour tout le monde, voici la donnée ALPHA, j'espère que vous l'aimerez »*
  - ✓ **Format Standard – spécification CAN 2.0A : 11 bits d'identification**
  - ✓ **Format Etendu – spécification CAN 2.0B : 29 bits d'identification**



- **Requêtes** *« Bonjour tout le monde, quelqu'un peut-il envoyer la donnée ALPHA ? »*
- **Surcharges** *« je vous ai entendu mais je suis occupé. Pouvez-vous attendre s'il vous plait ?»*
- **Erreurs** *[tout le monde crie] « Quiconque à envoyé ça, recommencer «*



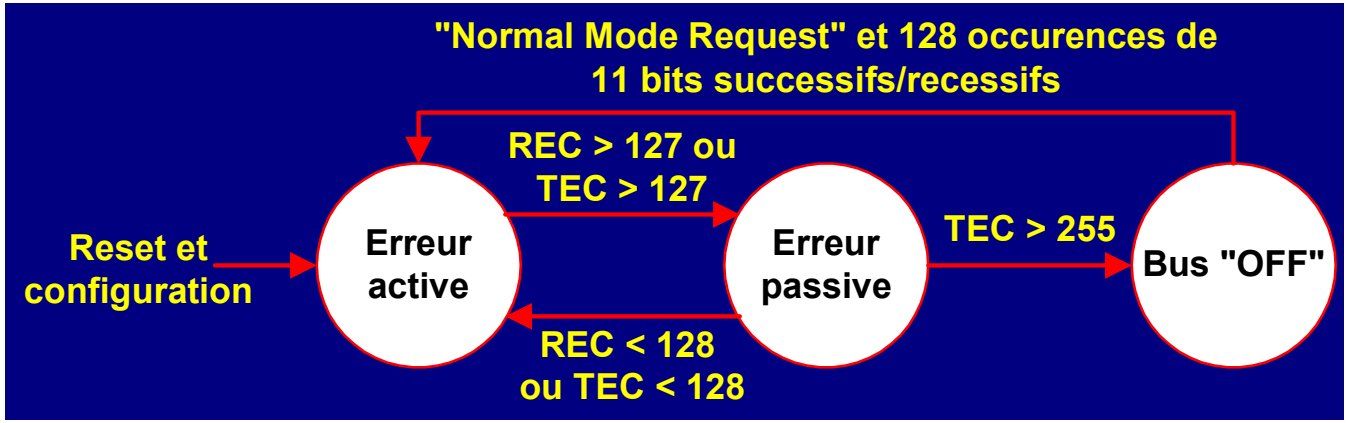
- ✓ Erreur Stuff-bit: 6 bits consécutifs de même niveau
- ✓ Erreur Bit : dominant + récessif = récessif
- ✓ Erreur CRC : CRC calculé <> CRC trame
- ✓ Erreur ACK : pas d'acquittement
- ✓ Erreur format : mauvaise valeur pour un champs fixe

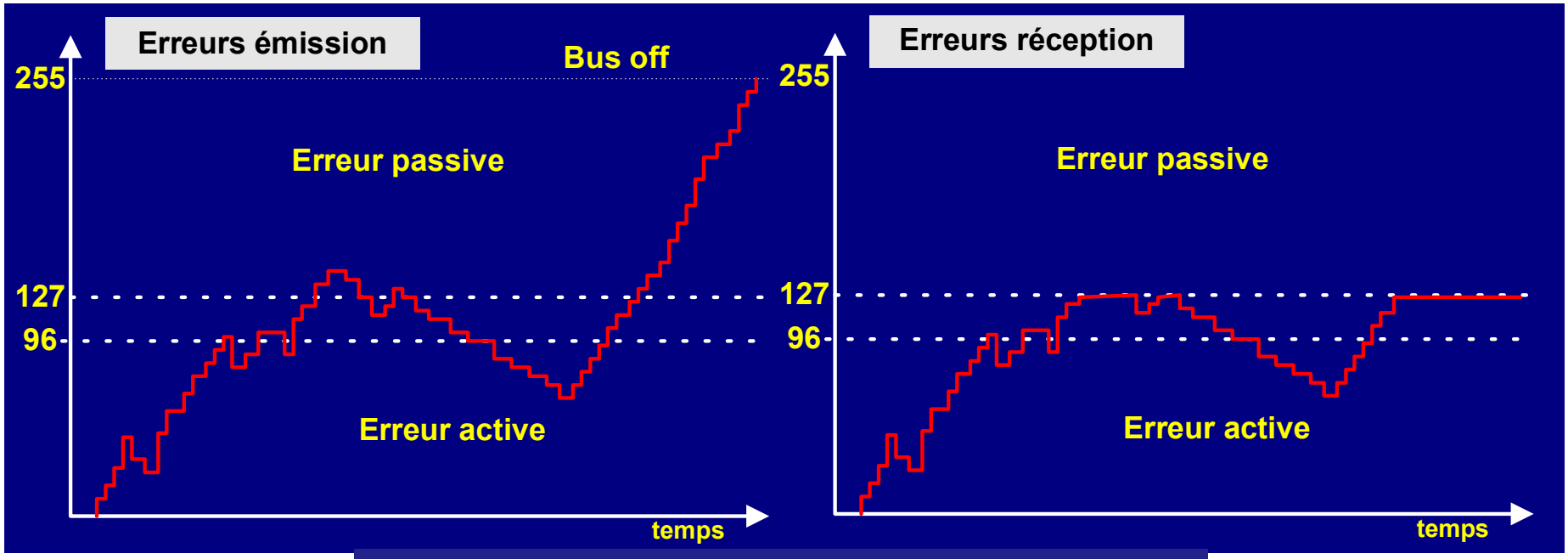
COMPTEURS

- ✓ **REC**
  - **Réception d'une trame corrompue: +1 (jusqu'à 128)**
  - **Réception d'une trame correcte: - 1 (si > 0)**
- ✓ **TEC**
  - **Emission d'une trame corrompue : +8 (jusqu'à 256)**
  - **Emission d'une trame correcte : -1 (si > 0)**

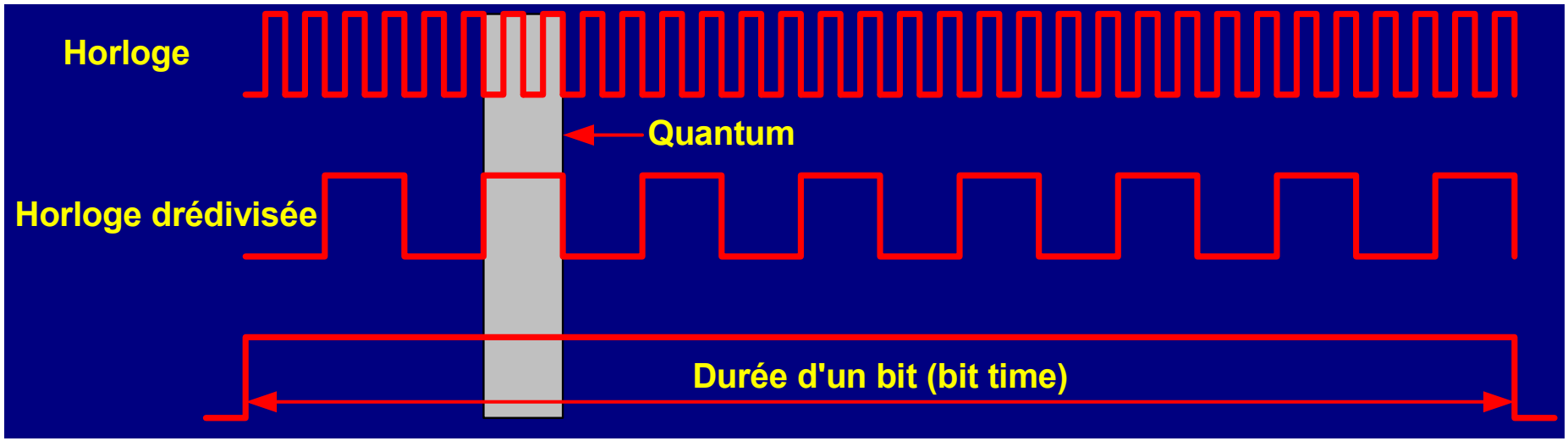
ETAT selon la valeur des compteurs

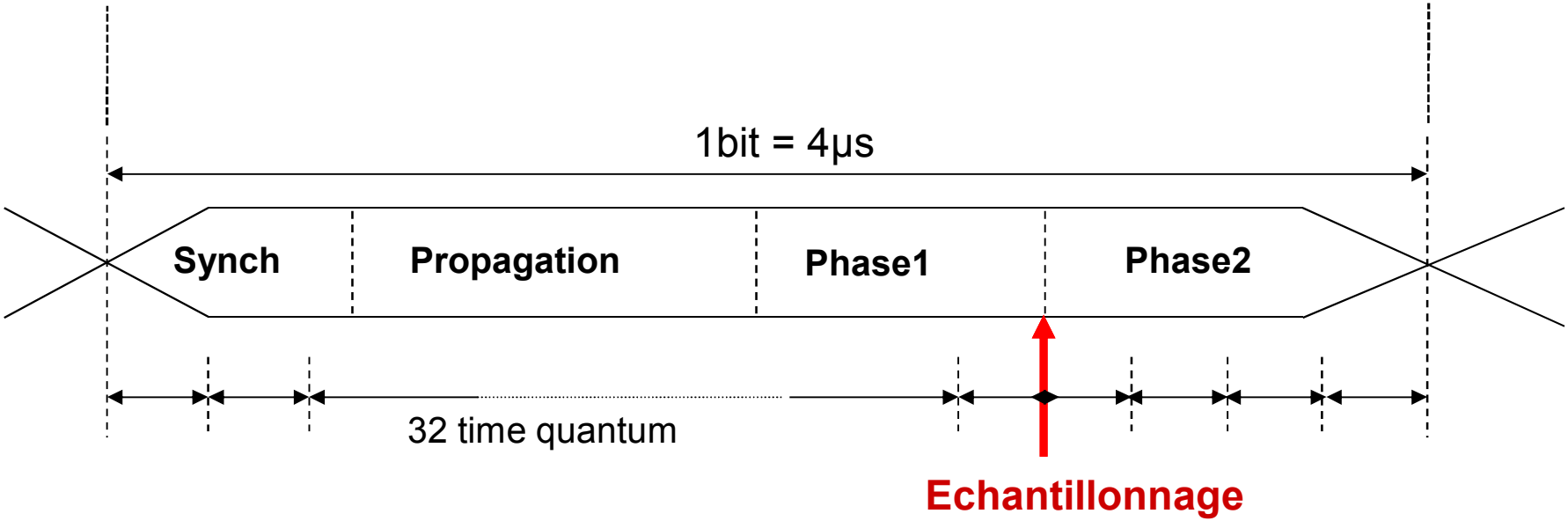
- ✓ **ERREUR-ACTIVE : fonctionnement normale**
- ✓ **ERREUR-PASSIVE : attente d'un temps mort de 6 bits récessifs sur le bus avant de réémettre**
- ✓ **BUS-OFF : nœud déconnecté du bus, reconnexion automatique ou logicielle possible**





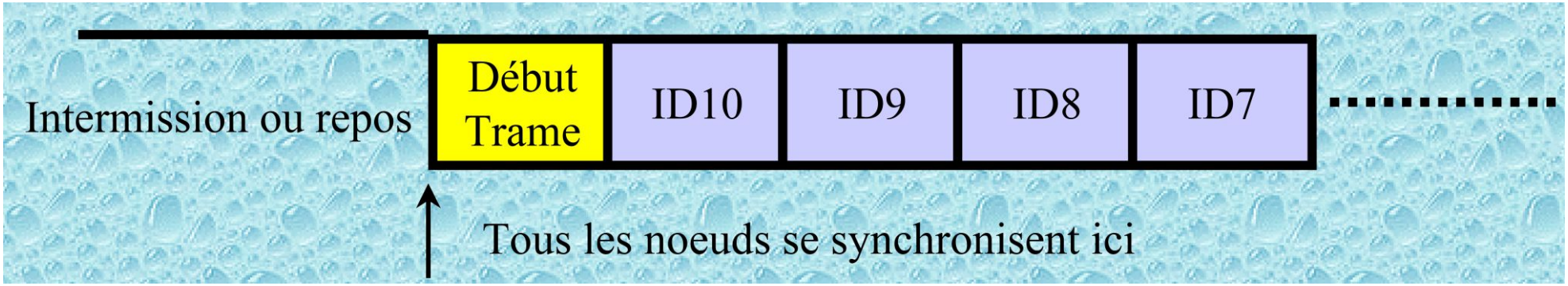
Un warning (interruption) est signalé à 96



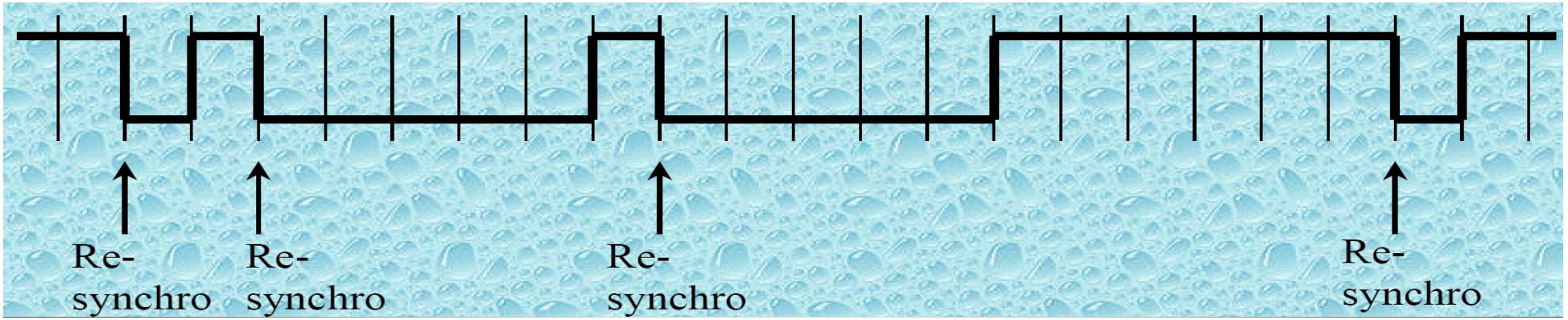


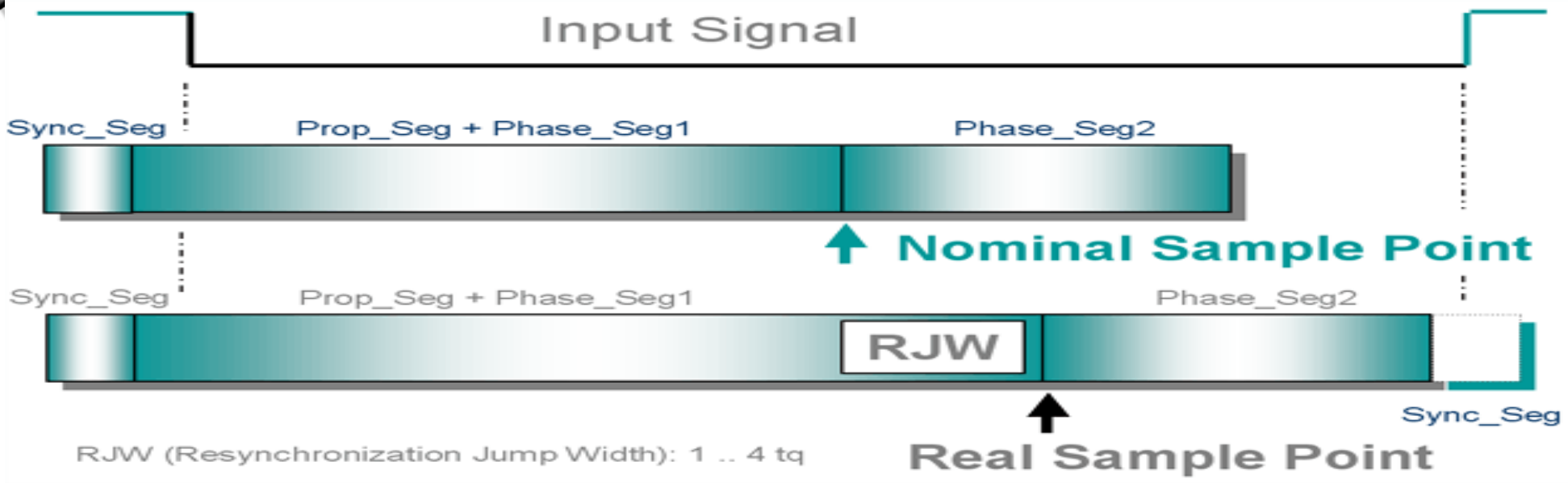
Vitesse de transmission ??

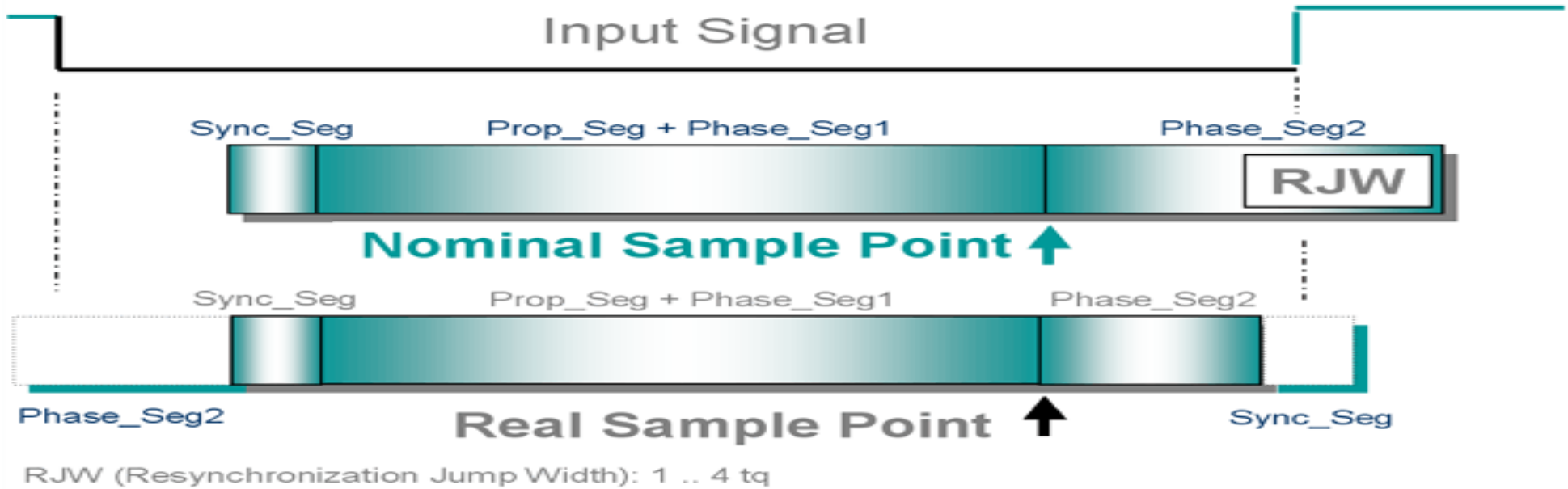
- **Hard synchro**

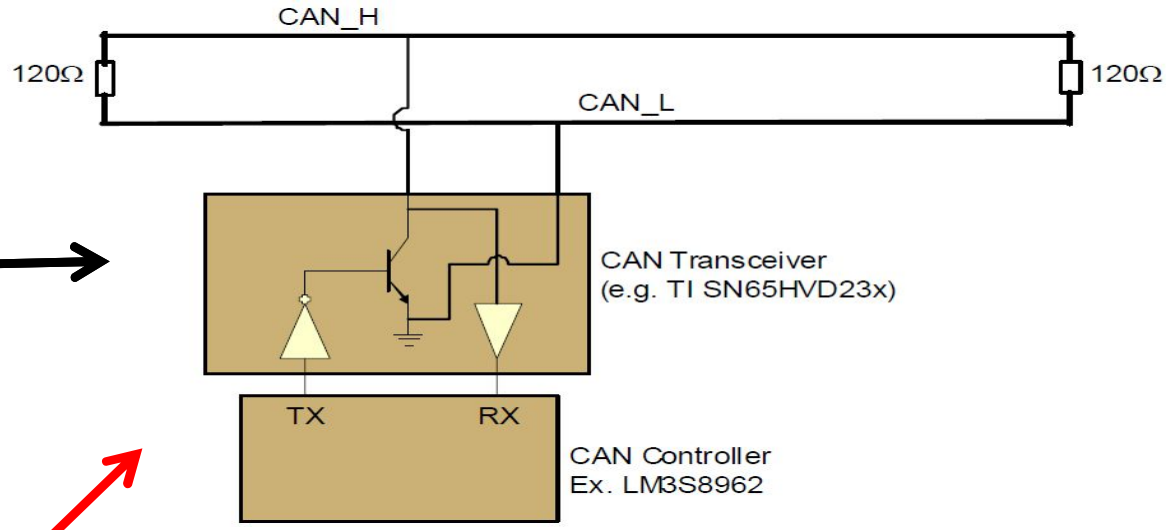


- **Re-synchro**

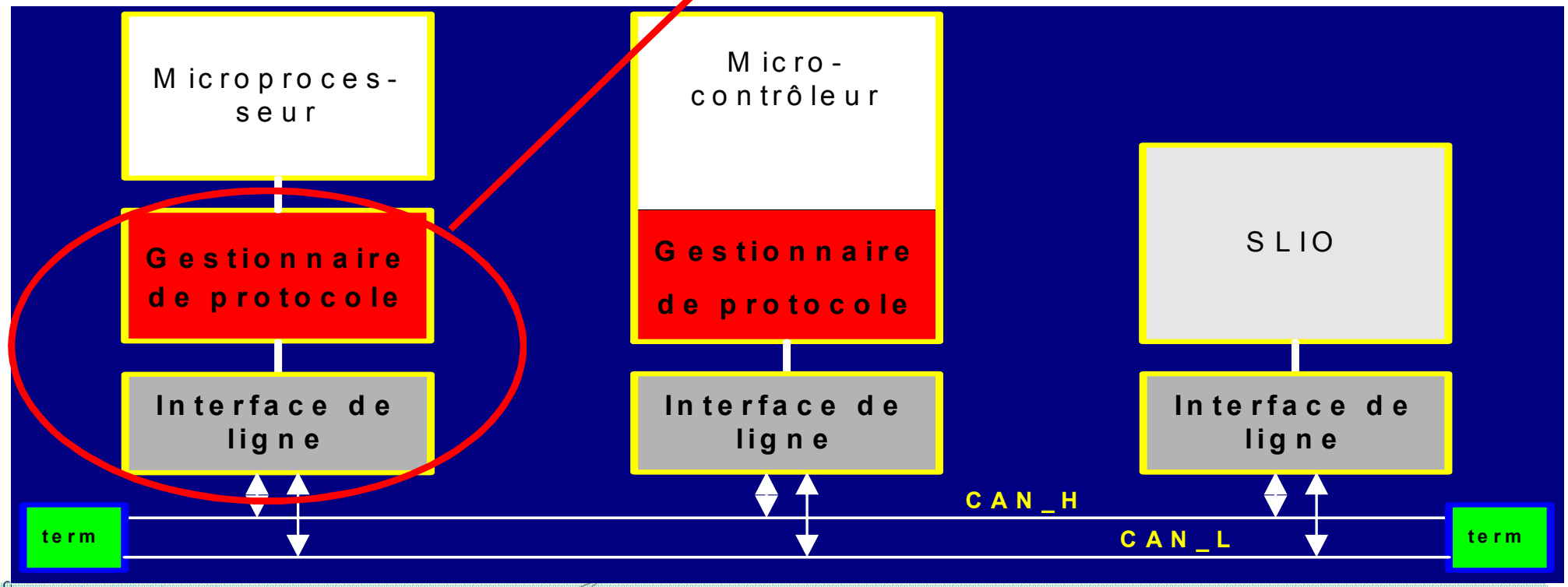


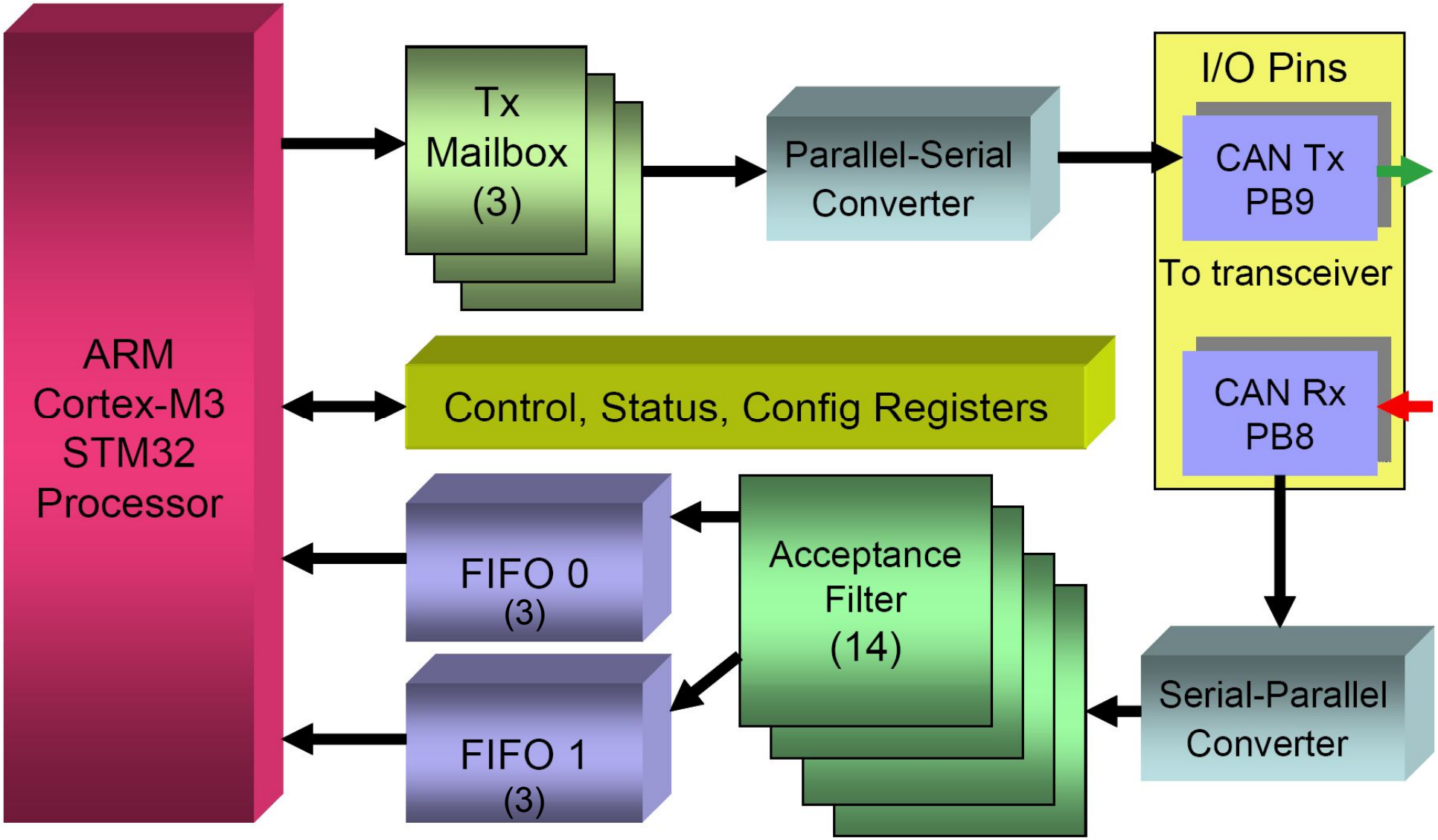

Input Signal
Sync_Seg
Prop_Seg + Phase_Seg1
Phase_Seg2
Nominal Sample Point
Sync_Seg
Prop_Seg + Phase_Seg1
Phase_Seg2
RJW
Sync_Seg
RJW (Resynchronization Jump Width): 1 .. 4 tq
Real Sample Point


Input Signal
Sync_Seg
Prop_Seg + Phase_Seg1
Phase_Seg2
RJW
Nominal Sample Point
Sync_Seg
Prop_Seg + Phase_Seg1
Phase_Seg2
Phase_Seg2
Real Sample Point
Sync_Seg
RJW (Resynchronization Jump Width): 1 .. 4 tq



Fixe le nombre de nœuds possibles




ARM
Cortex-M3
STM32
Processor
Tx
Mailbox
(3)
Parallel-Serial
Converter
I/O Pins
CAN Tx
PB9
To transceiver
CAN Rx
PB8
Control, Status, Config Registers
FIFO 0
(3)
FIFO 1
(3)
Acceptance
Filter
(14)
Serial-Parallel
Converter

MODE DE FILTRAGE

- ✓ 14 filtres de 2 registres 32-bit
- ✓ 4 modes de configuration
  - Taille 16 ou 32 bit
  - Liste ou masque d'identificateur

3

32-bit filter – Id/Mask

Id	STID[10:0]	EXID[17:0]		IDE/RTR/0
Mask				

1

32-bit filter – Id/List

Id	STID[10:0]	EXID[17:0]		IDE/RTR/0
Id	STID[10:0]	EXID[17:0]		IDE/RTR/0

16-bit filter – Id/Mask

Id	STID[10:0]	IDE/RTR/0
Mask		
Id	STID[10:0]	IDE/RTR/0
Mask		

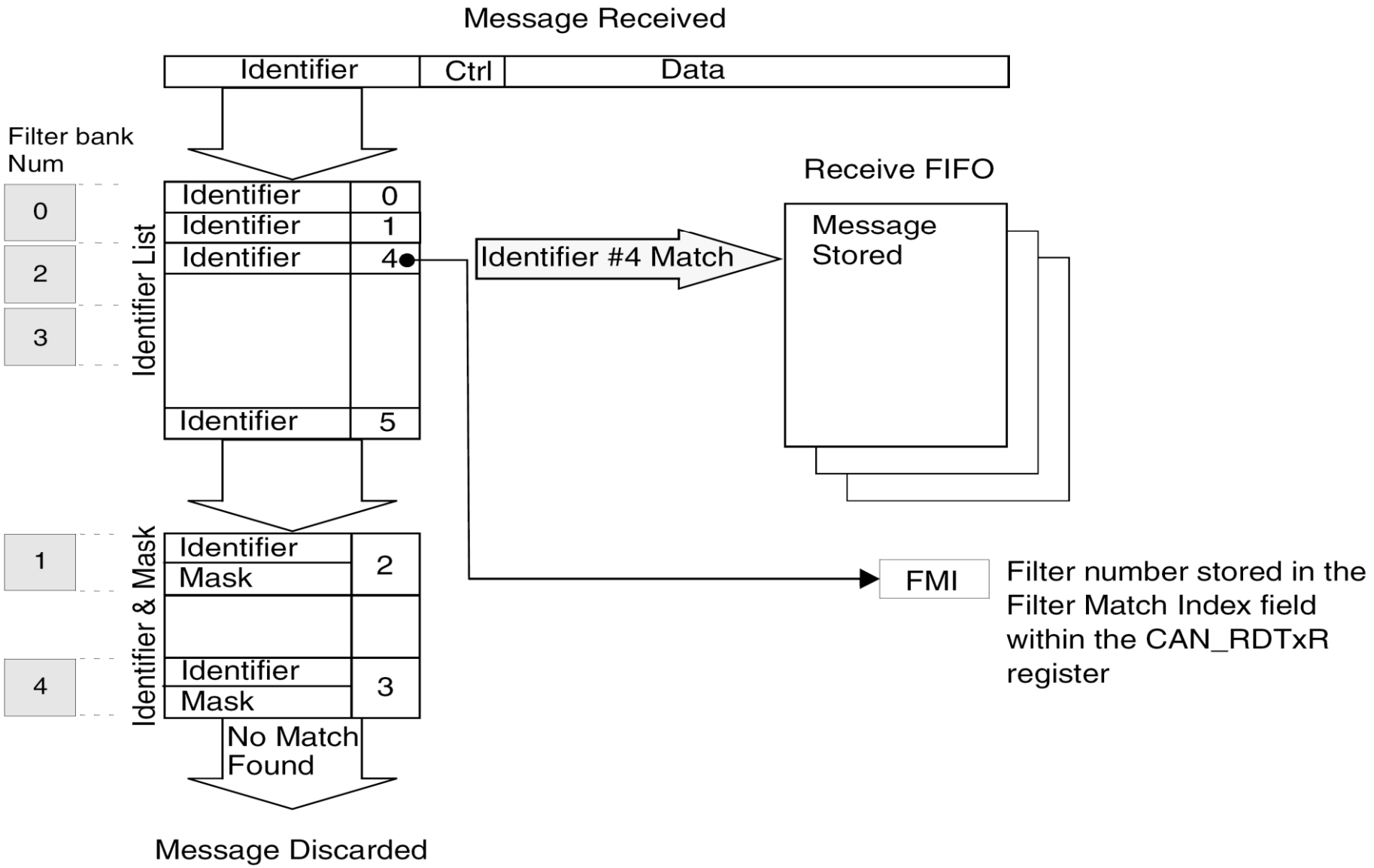
4

16-bit filter – Id/List

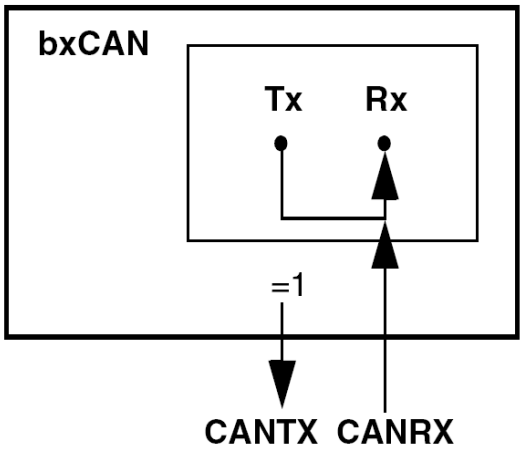
Id	STID[10:0]	IDE/RTR/0
Id	STID[10:0]	IDE/RTR/0
Id	STID[10:0]	IDE/RTR/0
Id	STID[10:0]	IDE/RTR/0

2

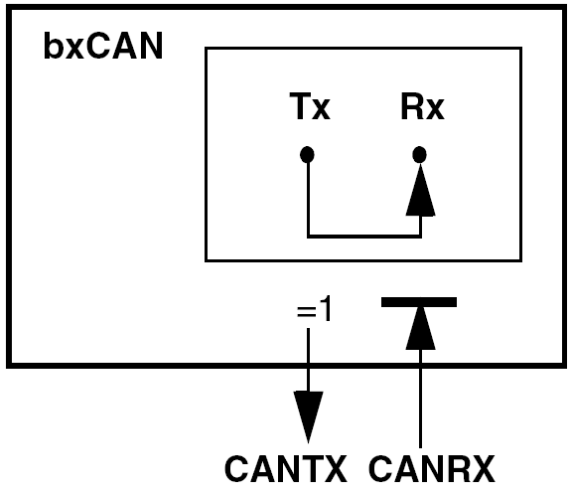
Example of 3 filter banks in 32-bit Unidentified List mode and the remaining in 32-bit Identifier Mask mode



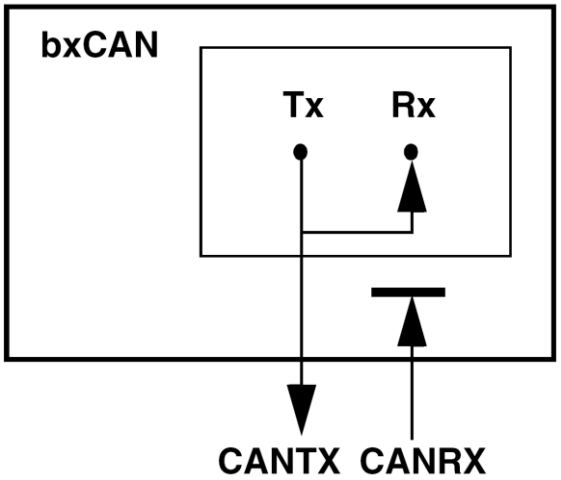
- **Normal**
- **Silent**

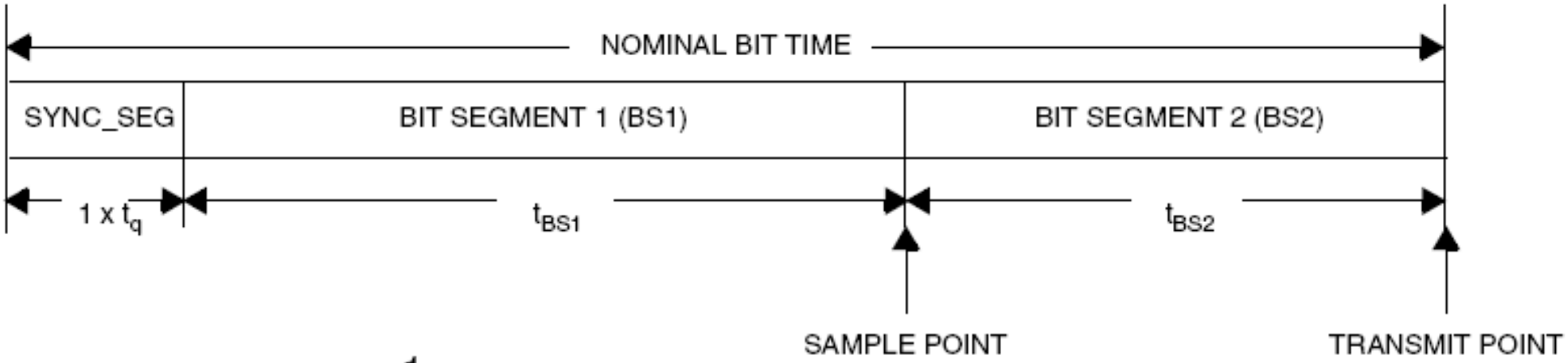


- **Mixte**



- **Loopback**





$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

with:

$t_{BS1} = t_q \times (TS1[3:0] + 1)$,

$t_{BS2} = t_q \times (TS2[2:0] + 1)$,

$t_q = (BRP[9:0] + 1) \times t_{PCLK}$

where t_q refers to the Time quantum

t_{PCLK} = time period of the APB clock,

BRP[9:0], TS1[3:0] and TS2[2:0] are defined in the CAN_BTR Register.

Main

Validation horloge

Init
Mode time_triggered ou event_triggered
Bus off
Wake_up
Retransmission
Verrouillage Fifo réception
Priorité transmission

Init Timing
SWJ, BS1, BS2, Prescaler

Mode
Normal
Silent
LoopBack
Silent_LoopBack

Init Filtres de réception
List ou Mask,
32 ou 16 bit
FIFO 0 ou 1

Validation filtres

Init Message

Transmission message

mailbox vide ?

Réception FIFO ?

Reception Message

```
/* CAN cell init */
 CAN_InitStructure.CAN_TTCM=DISABLE;        // Time Trigged Communication Mode
 CAN_InitStructure.CAN_ABOM=DISABLE;        // Automatic Bus-Off Management
 CAN_InitStructure.CAN_AWUM=DISABLE;        // Atomatic Wake-Up Mode
 CAN_InitStructure.CAN_NART=DISABLE;        // No-Automatic Retransmission mode
 CAN_InitStructure.CAN_RFLM=DISABLE;        // Receive Fifo Locked Mode
 CAN_InitStructure.CAN_TXFP=DISABLE;        // Transmit Fifo Priority
 CAN_InitStructure.CAN_Mode=CAN_Mode_LoopBack;
 CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
 CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
 CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
 CAN_InitStructure.CAN_Prescaler=45;        Ft = ?
 CAN_Init(&CAN_InitStructure);
```

```
/* CAN filter init */
 CAN_FilterInitStructure.CAN_FilterNumber=0;                    // N° du filtre 0 à 13
 CAN_FilterInitStructure.CAN_FilterMode=CAN_FilterMode_IdList;
 CAN_FilterInitStructure.CAN_FilterScale=CAN_FilterScale_16bit;
 CAN_FilterInitStructure.CAN_FilterIdHigh=0x0220;       // identificateur ??
 CAN_FilterInitStructure.CAN_FilterIdLow=0x0000;
 CAN_FilterInitStructure.CAN_FilterMaskIdHigh=0x0000;
 CAN_FilterInitStructure.CAN_FilterMaskIdLow=0x0000;
 CAN_FilterInitStructure.CAN_FilterFIFOAssignment=0;            // FIFO réception 0 ou 1
 CAN_FilterInitStructure.CAN_FilterActivation=ENABLE;
 CAN_FilterInit(&CAN_FilterInitStructure);
```

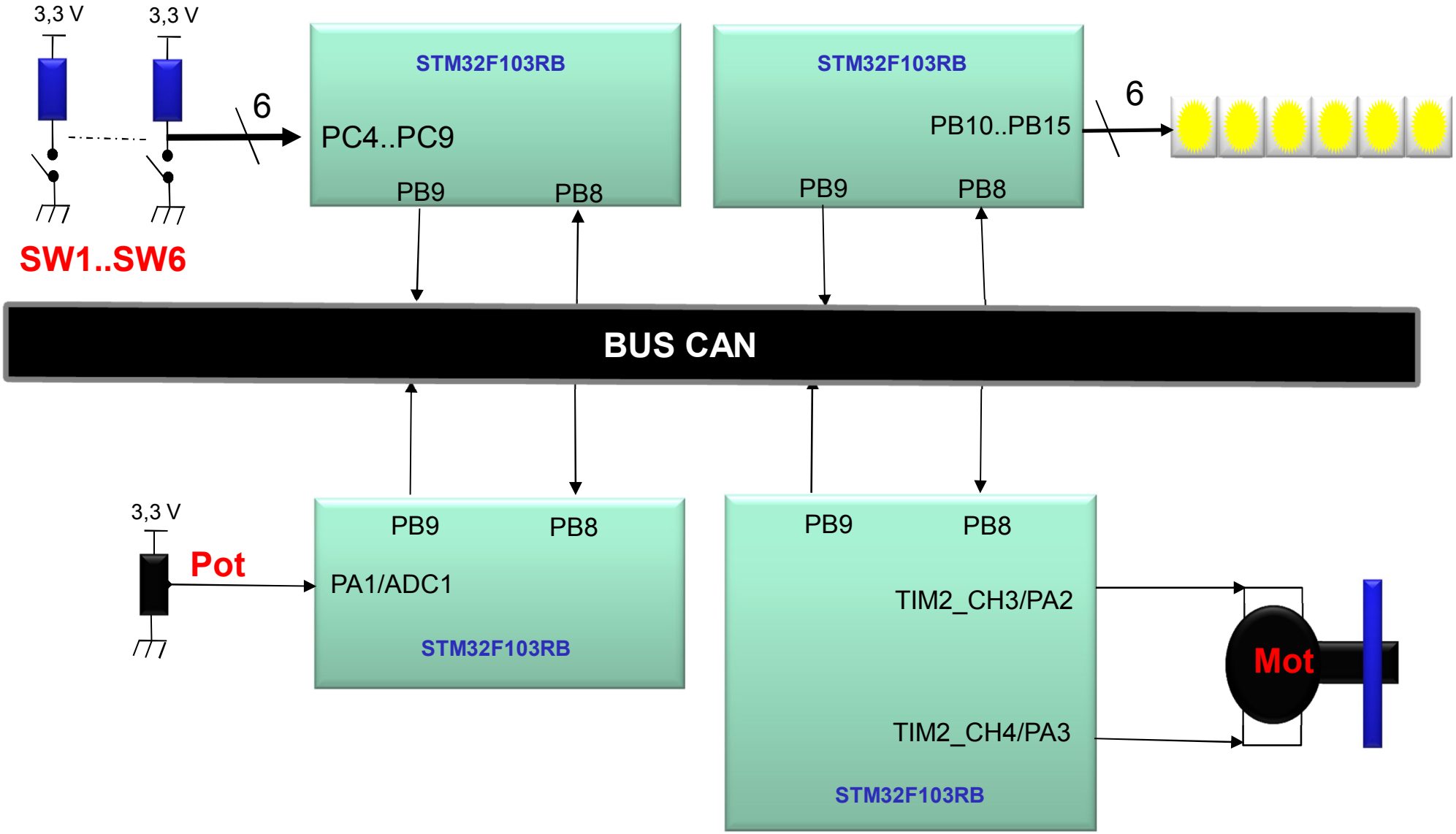
```
/* transmit */
TxMessage.StdId=0x11;                    // Identificateur
TxMessage.RTR=CAN_RTR_DATA;              // Data Frame ou remote frame
TxMessage.IDE=CAN_ID_STD;                // identificateur standard ou étendu
TxMessage.DLC=2;                         // nombre d'octet envoyé
TxMessage.Data[0]=0x12;
TxMessage.Data[1]=0x34;
TransmitMailbox=CAN_Transmit(&TxMessage);
while(CAN_TransmitStatus(TransmitMailbox) != CANTXOK) ; //transmission terminée?
while(CAN_MessagePending(CAN_FIFO0) < 1) ;          // Message reçu ??
/* receive */
CAN_Receive(CAN_FIFO0, &RxMessage);
message = (RxMessage.Data[0]<<8)|RxMessage.Data[1];
```

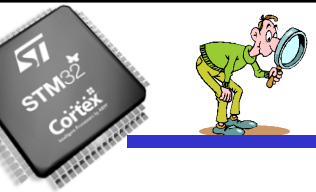
- Comment sont gérés les conflits sur le bus can multi-maitre ?
- Intérêt des compteurs d'erreur REC et TEC ?
- Sur un bus can à 500 kbit/s, quel débit utile (données) peut-on espérer ?
- Quelle est la durée maximale de transmission d'une trame can sur un réseau à 125 kbit/s ?
- Combien de mailboxes de transmission sur le STM32 ?
- Quels sont les interruptions possibles sur le bus can du STM32 ?

```
*-----------------------------------------------------------------------------------------------------------
* FONCTION :  Transfert d'une donnée de 2 octets 0x1234 via le bus can en mode
*                    loopback à 100 kbit/s avec RJW = 1 tq, BS1 = 8 tq et BS2 = 7 tq
*                    Le can sera géré par polling, ID = 0x11.
*
*              Utilisez le debugger pour vérifier la transmission correcte de la donnée
*-----------------------------------------------------------------------------------------------------------

*-----------------------------------------------------------------------------------------------------------
* FONCTION :  Améliorer le programme précédent en gérant le can par interruption en
*                    transmission et en réception.
*
*              Utilisez le debugger pour vérifier la transmission correcte de la donnée
*-----------------------------------------------------------------------------------------------------------
```

```
*-------------------------------------------------------------------------------------------------------------
* FONCTION :  Connecter 4 kits via leur bus CAN.
*
*                 Le premier kit doit envoyer l'état des dip-switches SW1 à SW6
*                 Le second doit afficher l'état de ces switches sur les leds PB10 à PB15
*                 Le troisième doit envoyer la valeur numérique issue du potentiomètre
*                 Le quatrième doit commander le moteur en PWM à partir de cette
*                valeur numérique
*-------------------------------------------------------------------------------------------------------------
```

<u>Conseil</u>: Faire fonctionner chaque kit en mode loopback en simulation et sur la cible puis passer en mode Normal


3,3 V
3,3 V
6
SW1..SW6
STM32F103RB
PC4..PC9
PB9
PB8
STM32F103RB
PB10..PB15
PB9
PB8
6
BUS CAN
3,3 V
Pot
PB9
PB8
PA1/ADC1
STM32F103RB
PB9
PB8
TIM2_CH3/PA2
TIM2_CH4/PA3
STM32F103RB
Mot



END

Directed by Acacio MARQUES & Co