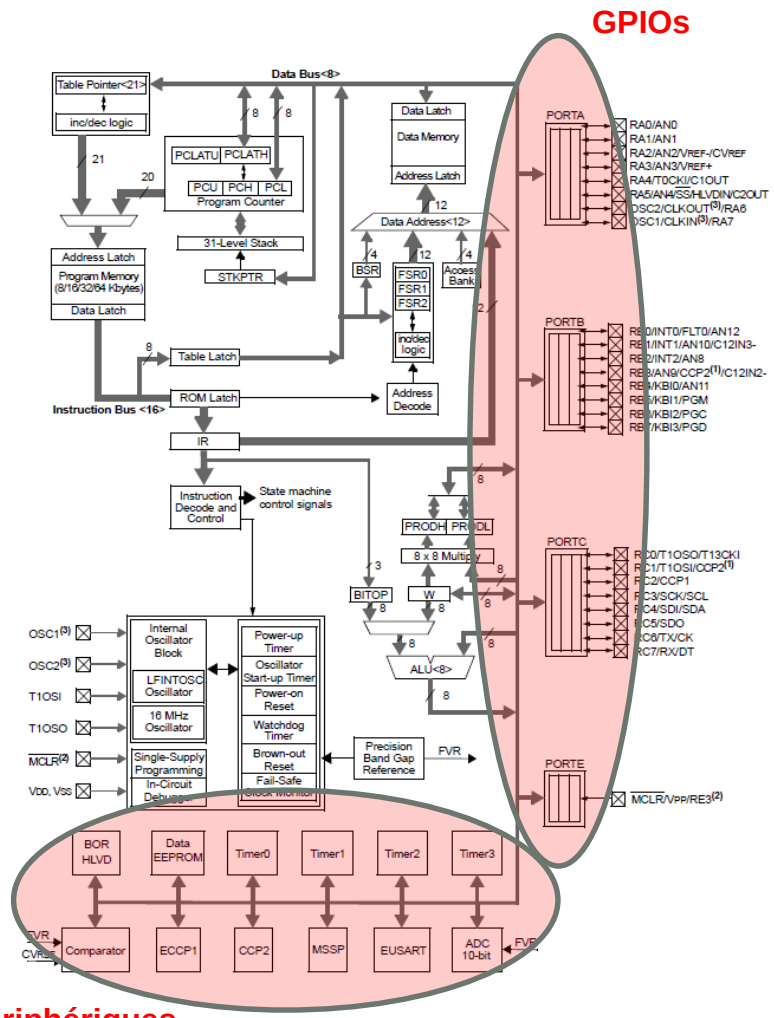
Architecture des microcontrôleurs

les entrées-sorties et les périphériques

Sommaire

1. Vue d'ensemble



Les microcontrôleur doit être capable d'interagir avec son environnement extérieur

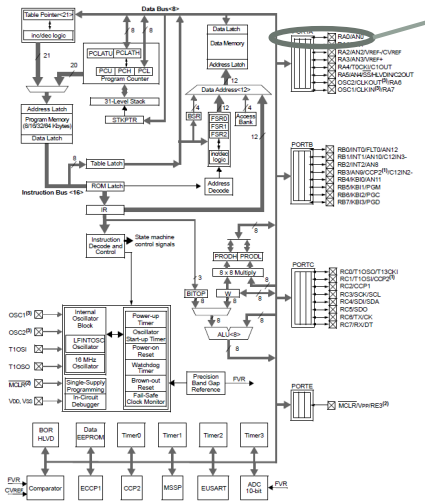
- pour recevoir des informations sur ses **entrées**, (ou **input**)
- pour transmettre des informations sur ses **sorties** (ou **output**).

Pour des informations tout-ou-rien classiques, on utilise des pattes d'entrées-sorties (E/S ou I/O) simples (**GPIO**: **G**eneral **P**urpose **I**nput **O**utput)

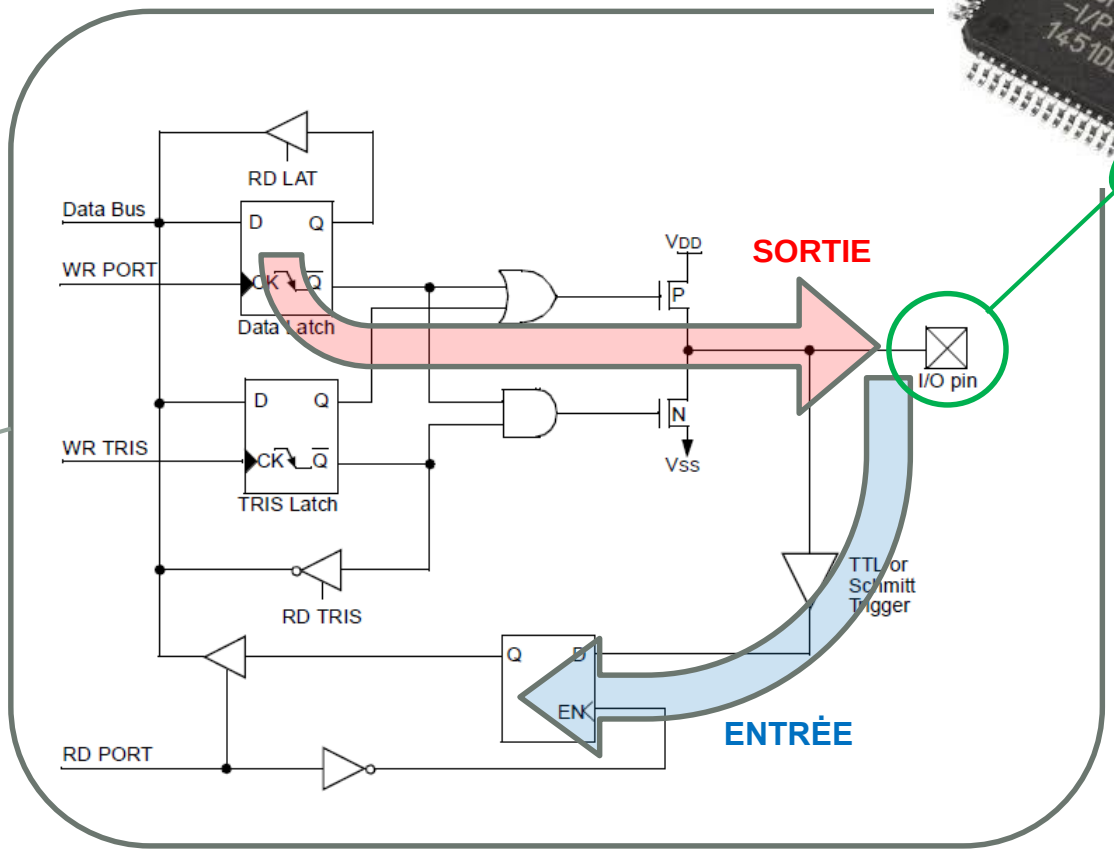
Pour des informations plus sophistiquées, on utilise des pattes et des **périphériques** dédiés:

- convertisseur analogique–numérique (ADC),
- convertisseur numérique–analogique (DAC),
- timers (temporisation, mesure de temps, PWM),
- liaisons série (UART, I²C, SPI, USB, CAN, etc.),
- commande d'afficheur LCD,
- etc.

2. Les ports d'entrée-sortie



1 GPIO



Les pattes d'E/S sont regroupées par « **ports** ».

Sur un PIC18, un port regroupe généralement 8 E/S (port A, port B, etc.).

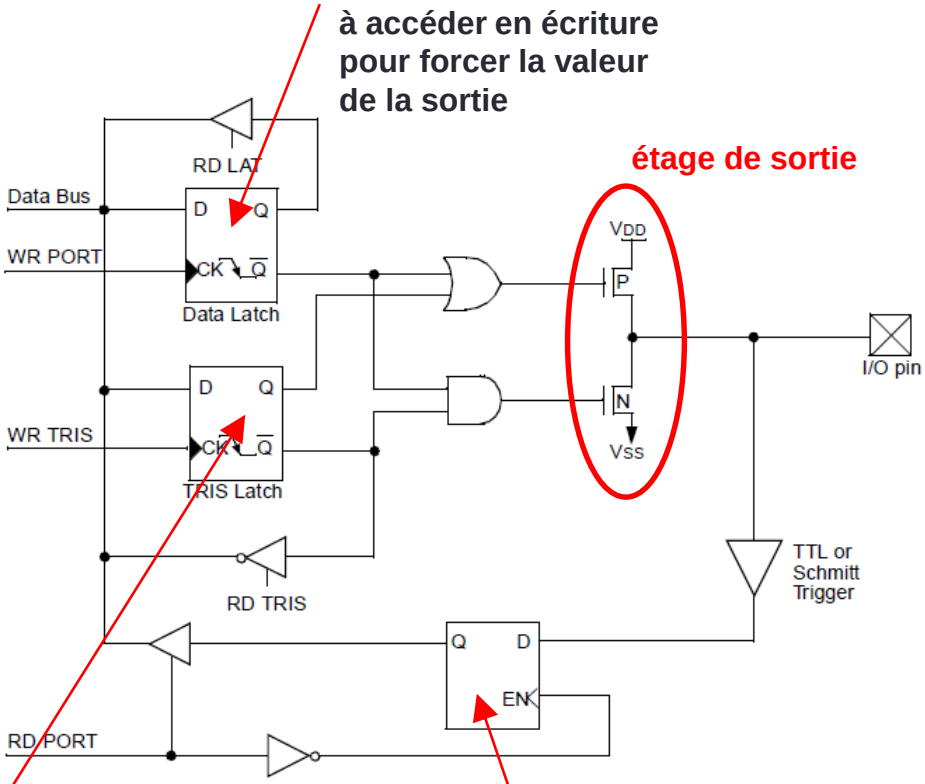
La majeure partie des E/S peuvent être configurées comme entrées **ou** comme sorties.

Pour chaque port, les pattes sont contrôlées par 3 registres dédiés:

- **TRIS** (direction)
- **LAT** (valeur en sortie)
- **PORT** (valeur en entrée)

Une entrée peut être utilisée comme source d'interruption.

Une E/S peut être associée à une fonction analogique (ADC ou DAC).



LAT
(valeur de la sortie)

à accéder en écriture pour forcer la valeur de la sortie

étage de sortie

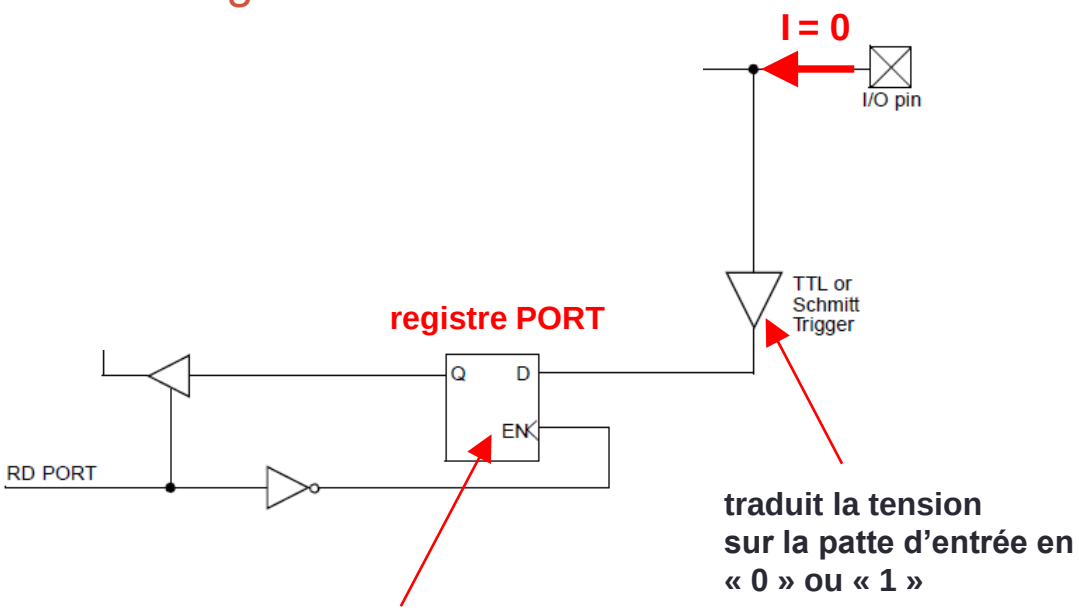
TRIS
(direction: active/désactive l'étage de sortie)

1: Input (comme 1nput)
0: Output (comme 0utput)

PORT
(valeur de l'entrée)

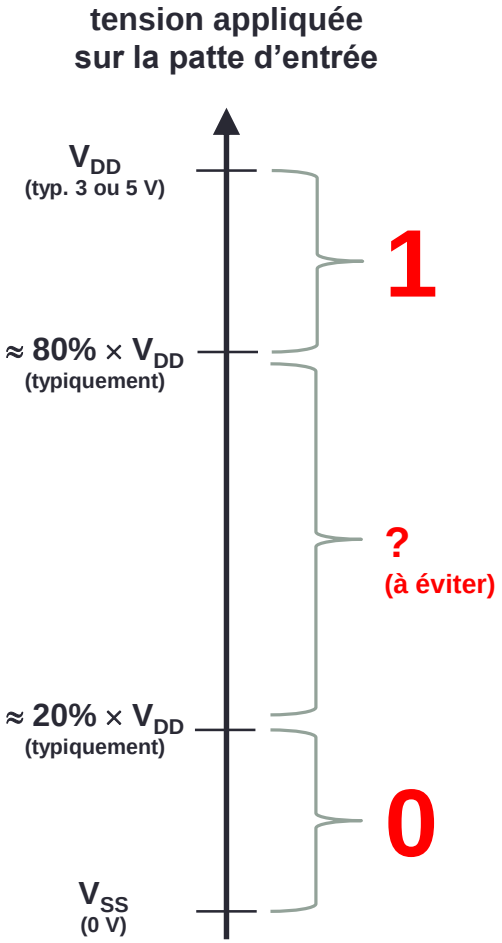
à accéder en lecture pour lire la valeur de l'entrée

2.1 L'étage d'entrée



traduit la tension sur la patte d'entrée en « 0 » ou « 1 »

contient la valeur logique de la tension appliquée sur la patte

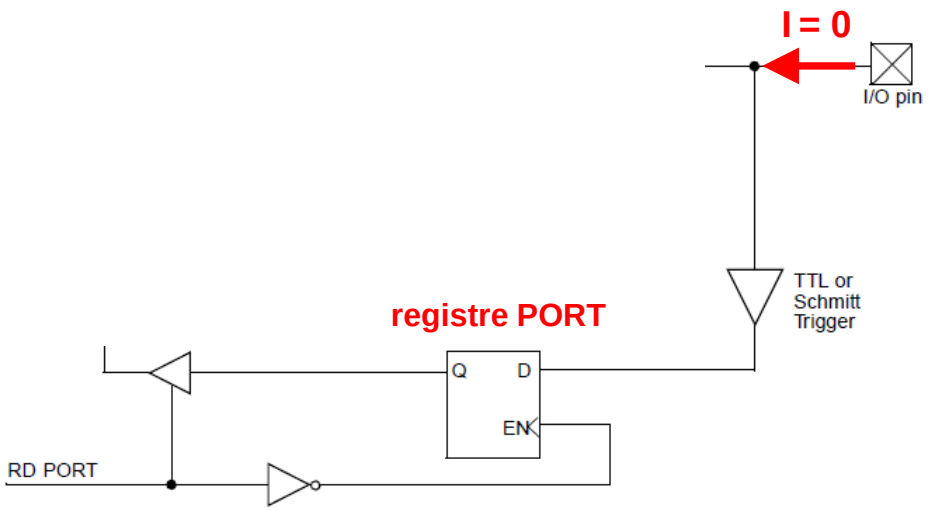


courant d'entrée ≈ 0 A

⇒ entrée HAUTE IMPÉDANCE

Remarque: **l'état de l'entrée doit toujours être fixé par « quelque chose » (la sortie d'un autre composant, un interrupteur, etc.).**

Cas d'une entrée flottante

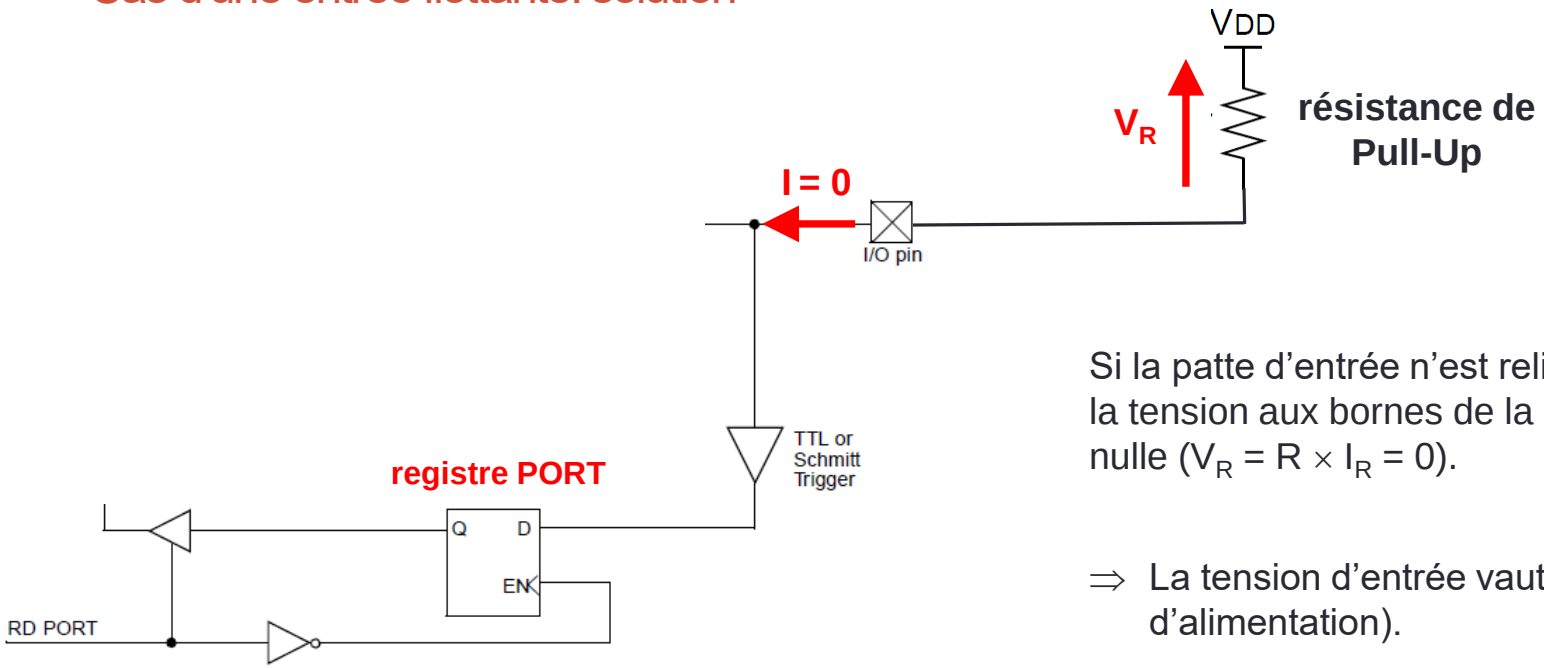


Si la patte d'entrée n'est reliée à rien (**entrée flottante**),
sa valeur logique n'est pas fixée.

⇒ Le contenu du registre PORT est aléatoirement 0 ou 1.

⇒ **À éviter !**

Cas d'une entrée flottante: solution



Si la patte d'entrée n'est reliée à rien d'autre, la tension aux bornes de la résistance est nulle ($V_R = R \times I_R = 0$).

⇒ La tension d'entrée vaut V_{DD} (tension d'alimentation).

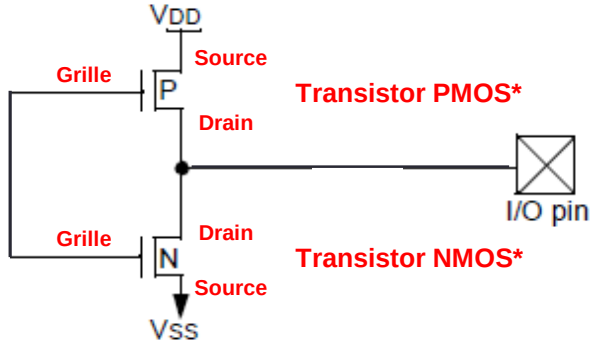
⇒ **La valeur d'entrée par défaut vaut 1.**

remarque: **la résistance de Pull-Up peut être interne ou externe au microcontrôleur.**

Sur certains microcontrôleurs, une résistance est disponible sur certaines pattes (peut être utilisée ou non).

PIC18: résistances de pull-up disponibles sur le port B, activables / désactivables individuellement (cf. bit RBPU dans le registre INTCON2, et registre WPUB)

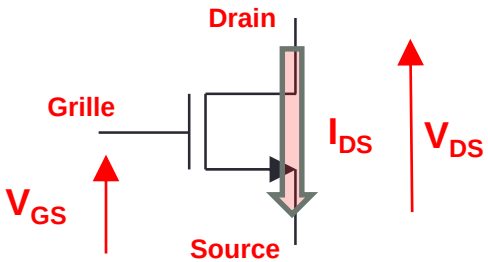
2.2 L'étage de sortie

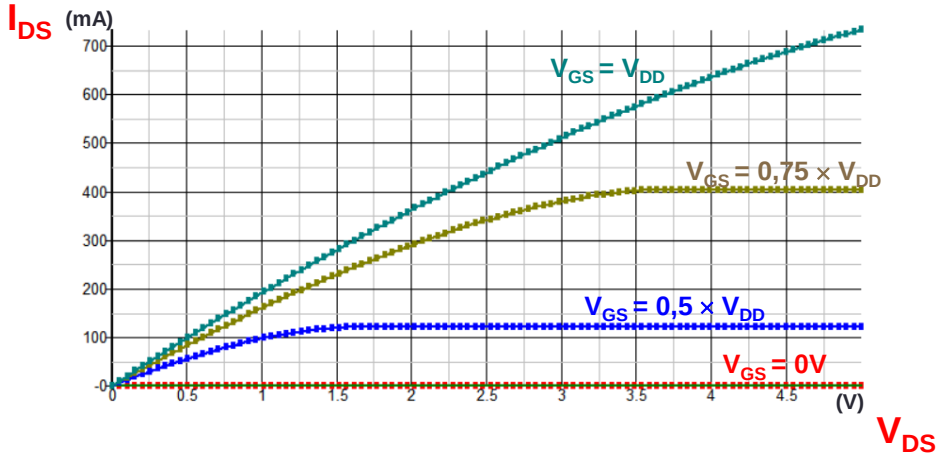


*: Metal Oxide Semiconductor

Le courant qui traverse le transistor (I_{DS})
est contrôlé par la tension entre la grille et la source (V_{GS}).

Remarque: ce courant I_{DS} est bien sûr aussi fonction de ce qui est connecté aux bornes du transistor.

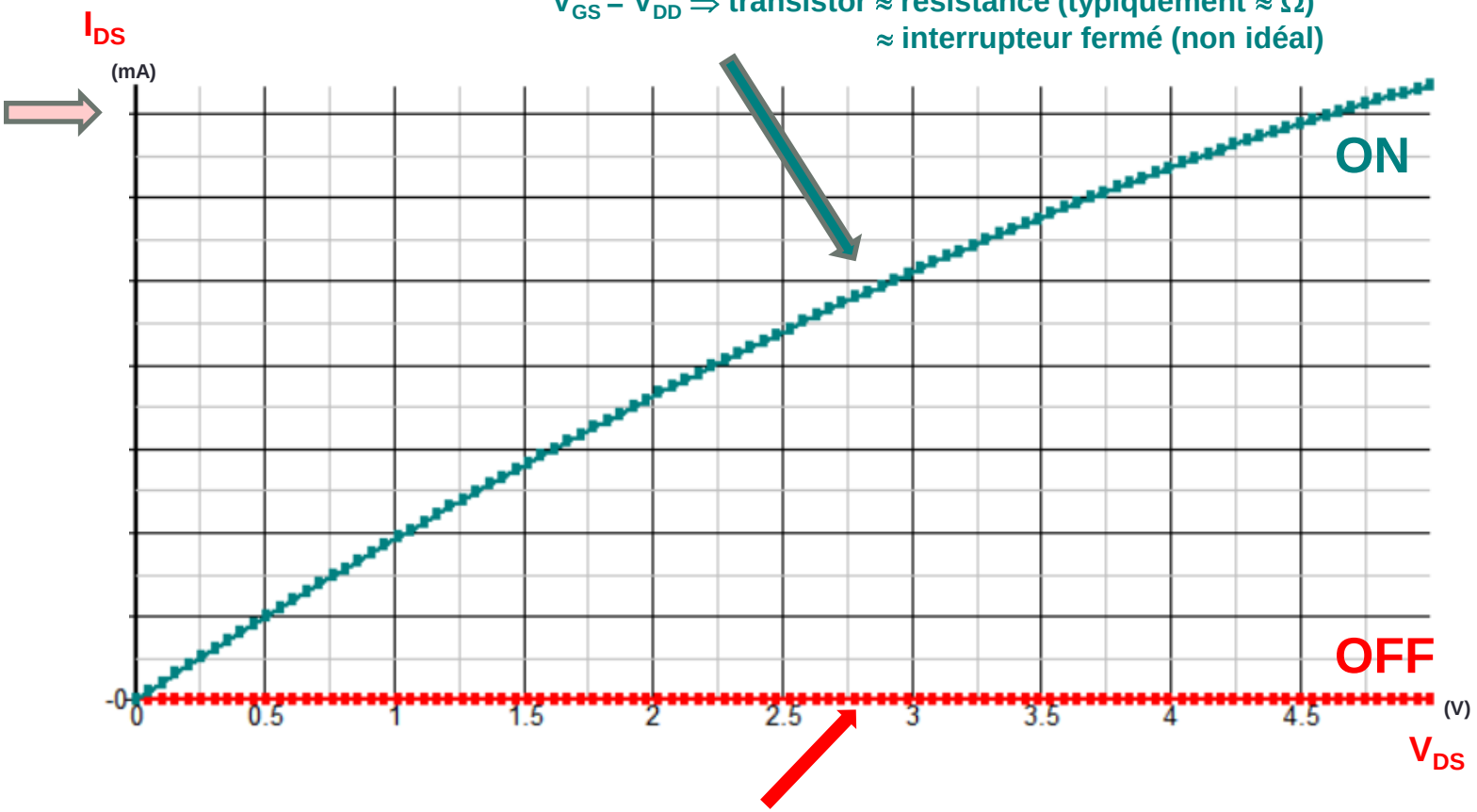




En électronique numérique, on ne rencontre que 2 cas extrêmes:

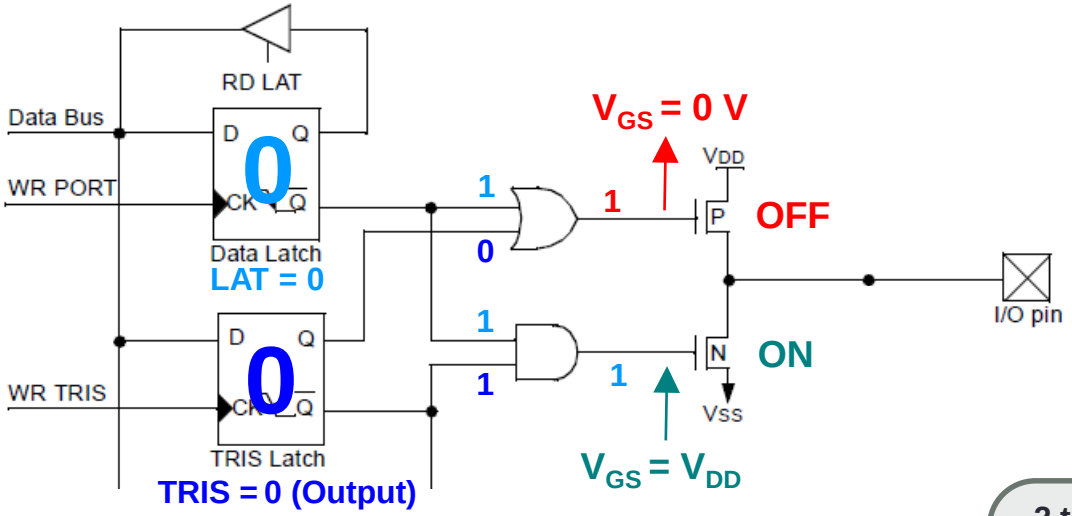
- V_{GS} = minimum = 0 V
- V_{GS} = maximum ≈ tension d'alimentation (notée V_{cc} ou V_{DD})

Typiquement 10 à 20 mA max pour un microcontrôleur



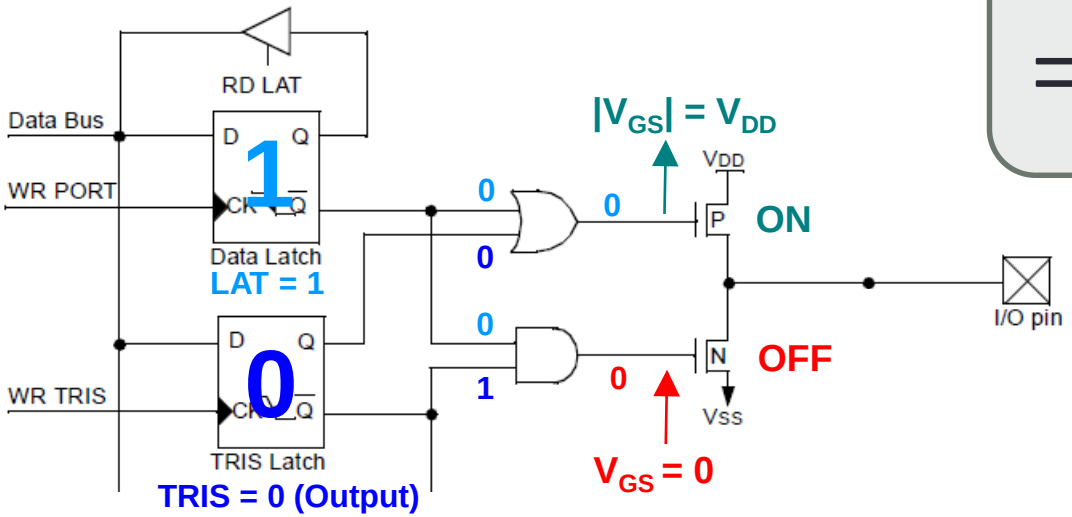
Retour sur l'étage de sortie

L'état logique chargé dans le registre LAT fixe l'état logique sur la patte d'Entrée/Sortie



0

tension ≈ V_{SS} (0 V)



1

tension ≈ V_{DD} (tension d'alimentation)

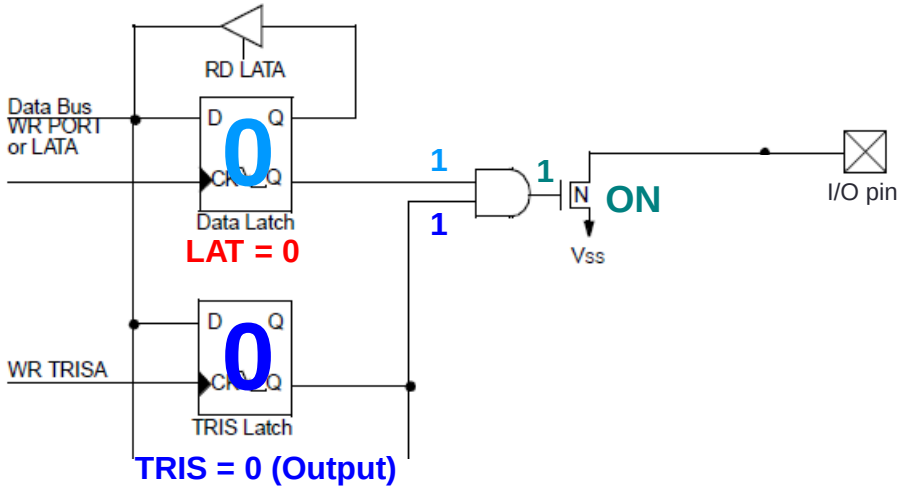
2 transistors complémentaires dans l'étage de sortie

⇒ **sortie PUSH-PULL**

(ou *basse impédance*)

remarque: si 1 est chargé dans le registre TRIS ⇒ les 2 transistors sont « OFF »

Variante possible de l'étage de sortie

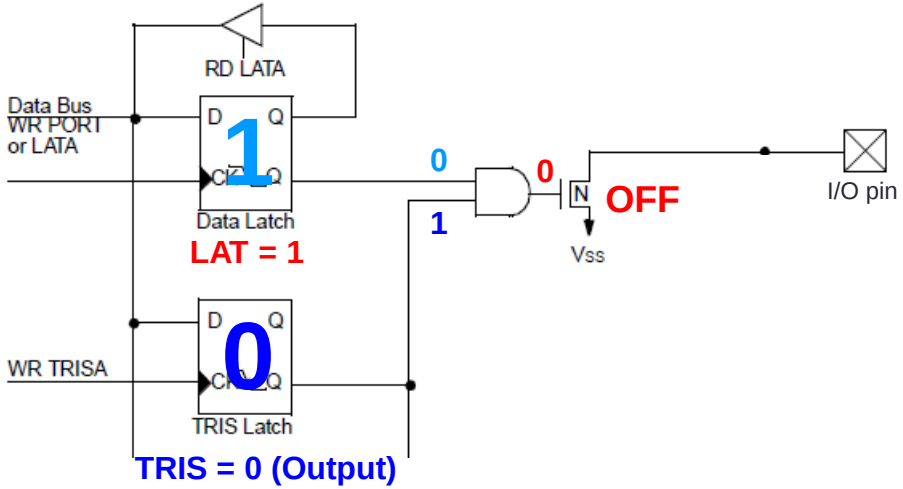


0

tension ≈ V_{SS} (0 V)

1 seul transistor dans l'étage de sortie

⇒ sortie OPEN DRAIN



?

tension indéterminée

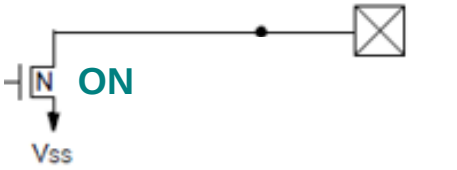
Sortie haute impédance (high Z)

mise à 0 de la sortie:

possible

0

tension ≈ 0 V

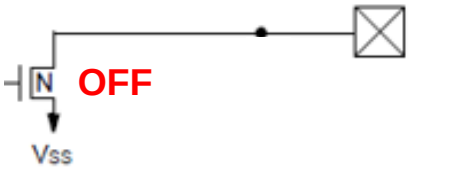


mise à 1 de la sortie:

impossible

0 ou 1?

tension = ?

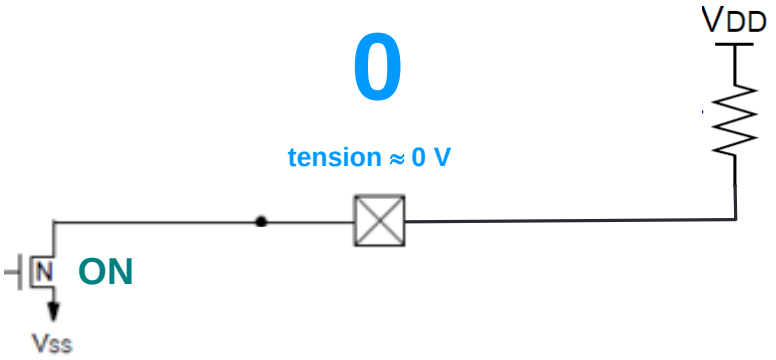


à moins d'ajouter une résistance fixant la sortie à 1 par défaut

(résistance de Pull-Up)

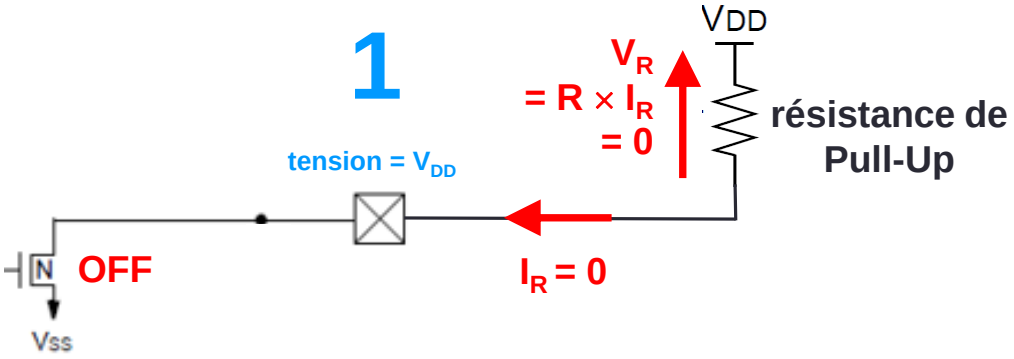
mise à 0 de la sortie:

possible



mise à 1 de la sortie:

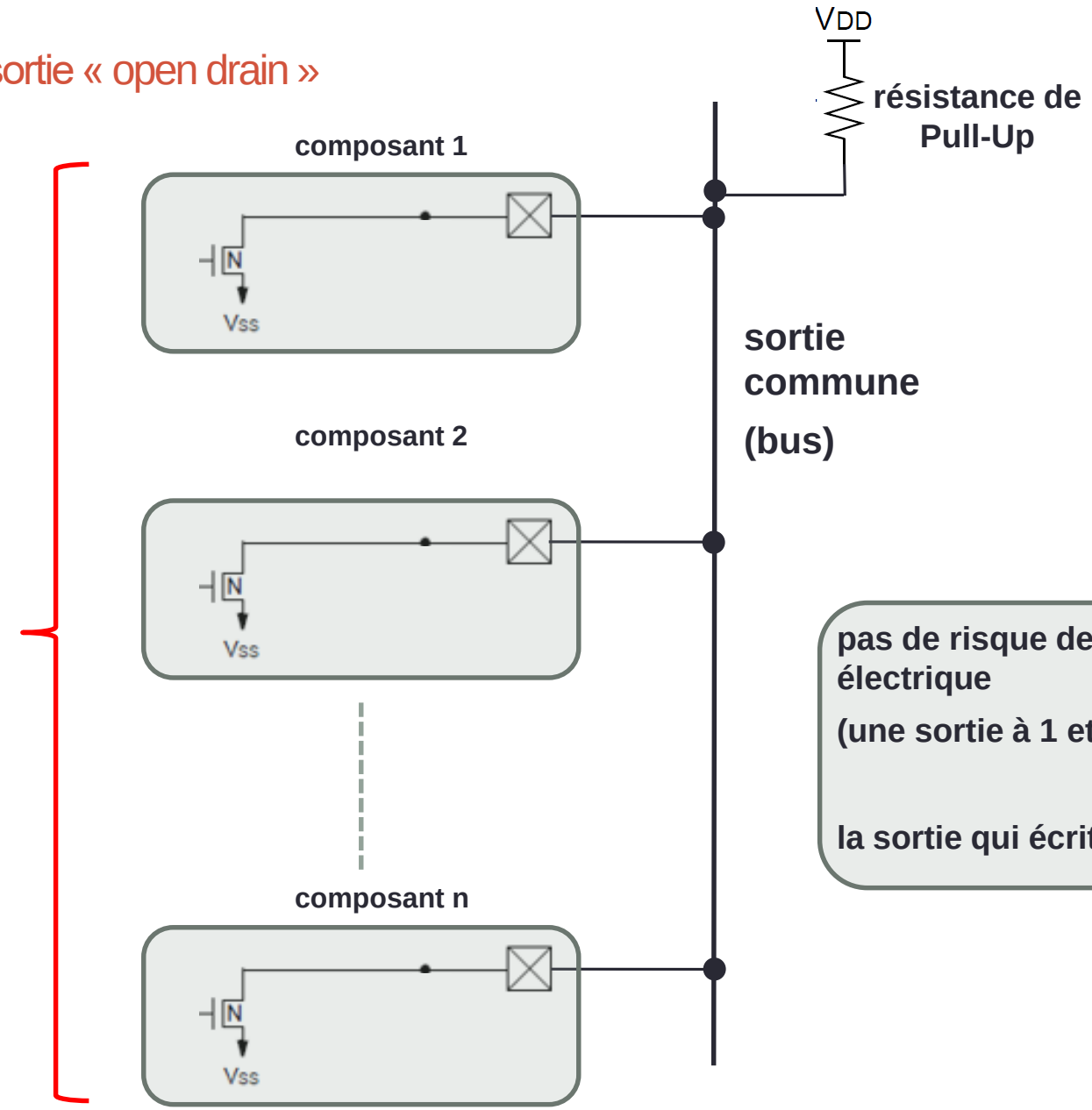
impossible



à moins d'ajouter une résistance fixant la sortie à 1 par défaut

(<u>résistance de Pull-Up</u>)

Intérêt d'une sortie « open drain »



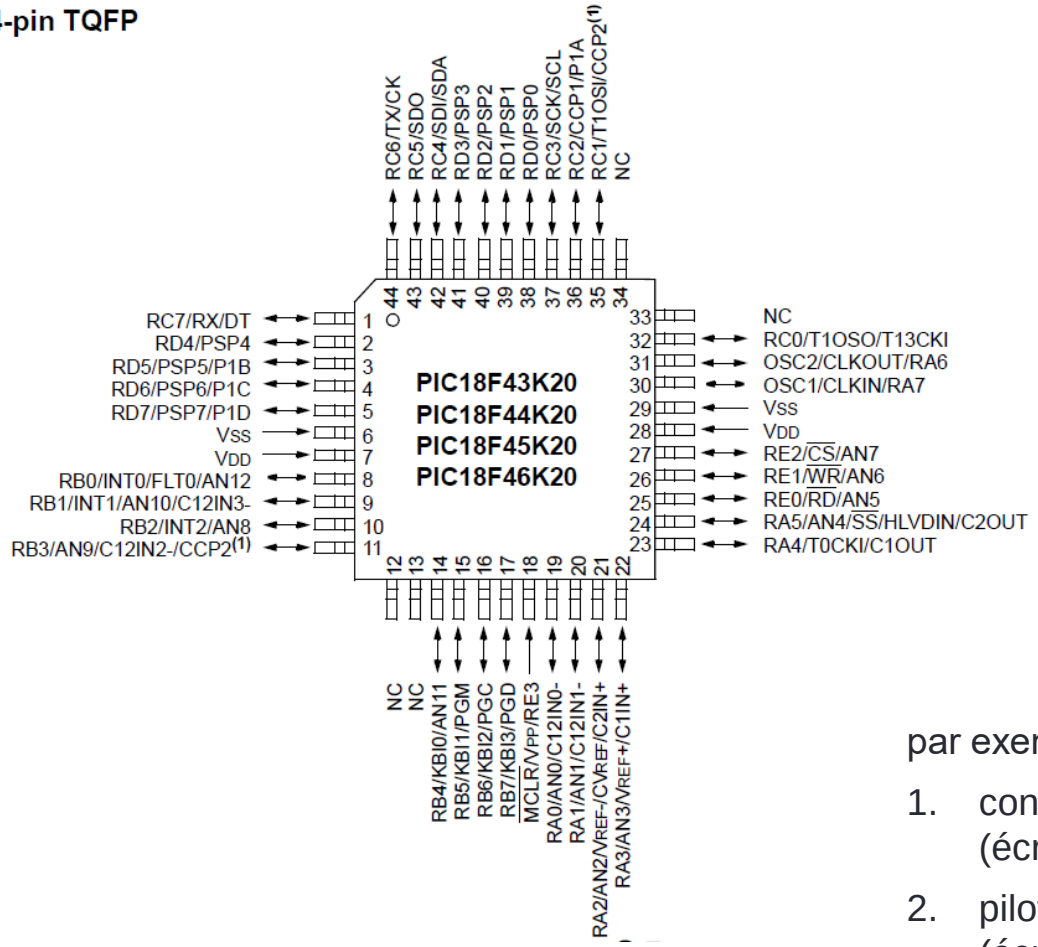
plusieurs sorties connectées sur une ligne commune

pas de risque de conflit électrique

(une sortie à 1 et une autre à 0)

la sortie qui écrit 0 « gagne »

Exemple du PIC18F45K20



1 port d'entrée/sortie **RA** (8 bits)

1 port d'entrée/sortie **RB** (8 bits)

1 port d'entrée/sortie **RC** (8 bits)

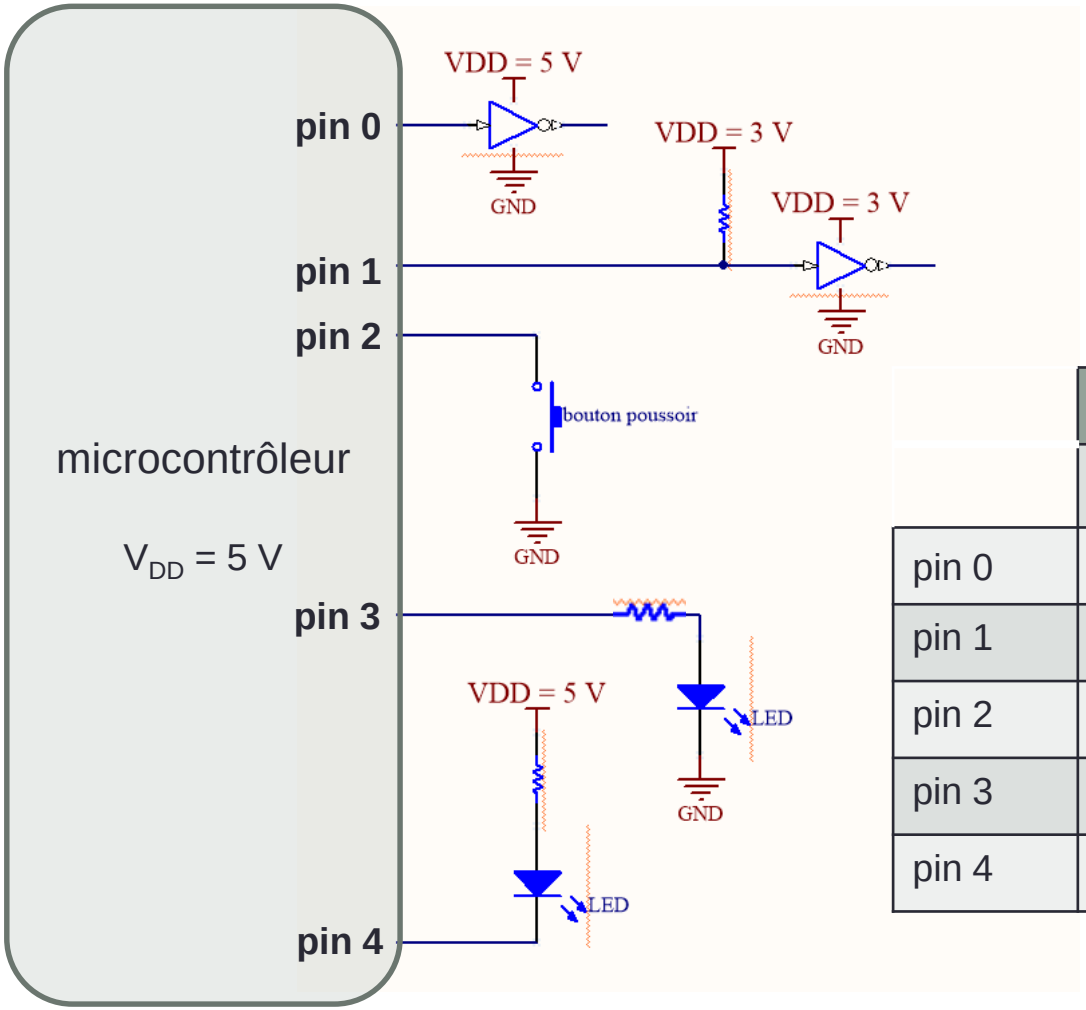
1 port d'entrée/sortie **RD** (8 bits)

1 port d'entrée/sortie **RE** (4 bits)

par exemple: pour forcer l'état sur la patte RA6

1. configurer le bit 6 du port A en sortie
   (écrire 0 dans le bit 6 du registre **TRISA**)
2. piloter la patte
   (écrire 0 ou 1 dans le bit 6 du registre **LATA**)

Quelles configurations dans les cas suivants?

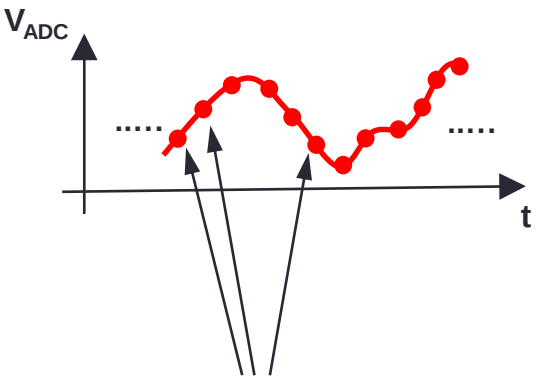


	IN		OUT	
	sans pull-up	avec pull-up	push-pull	open drain
pin 0				
pin 1				
pin 2				
pin 3				
pin 4				

3. Le convertisseur A/N (**A**nalog to **D**igital **C**onverter)

L'ADC convertit une grandeur analogique (la plupart du temps, une tension) en une valeur numérique codée sur plusieurs bits.

tension à l'entrée de l'ADC



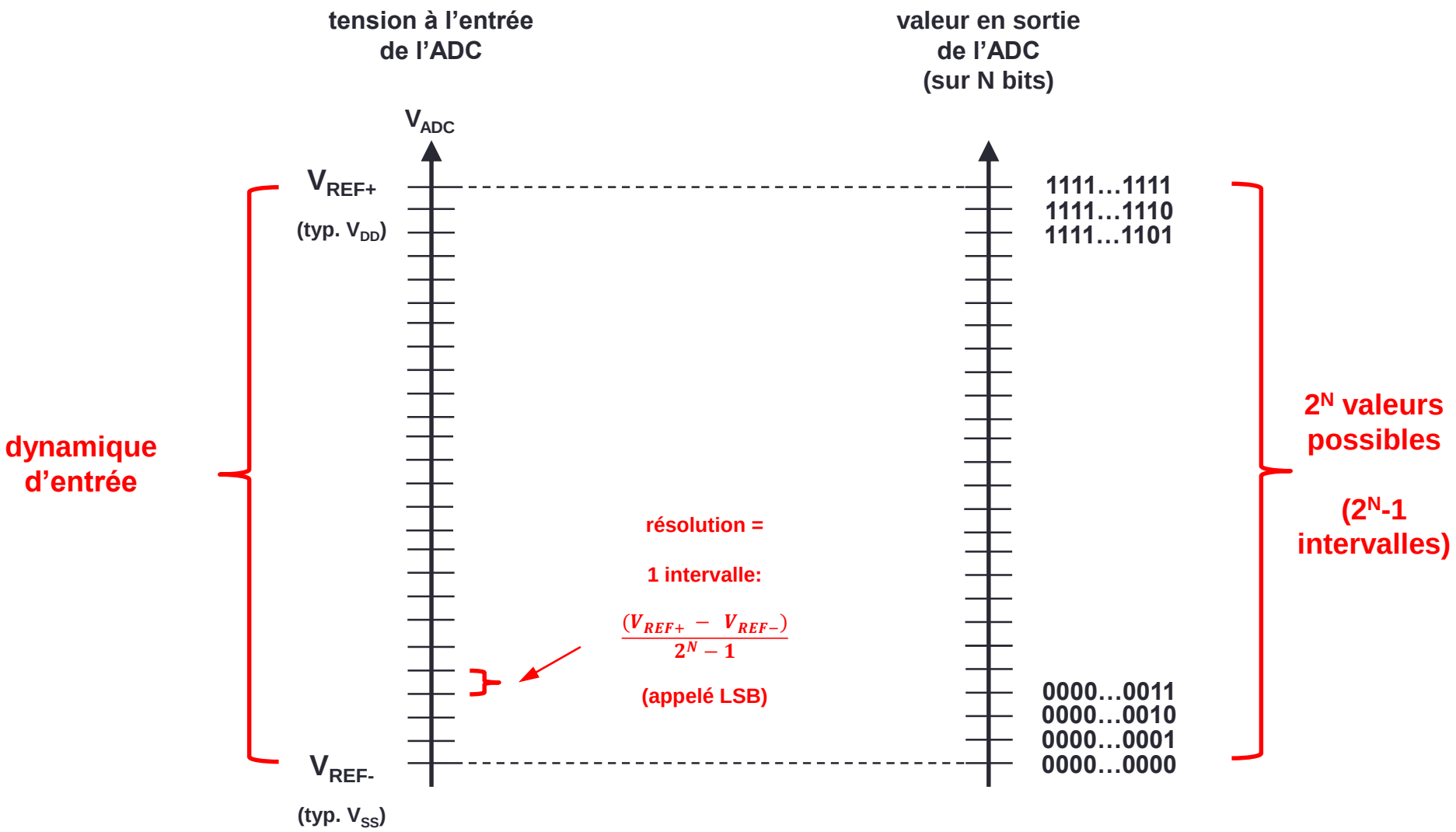
échantillons (samples)

valeurs numériques en sortie de l'ADC

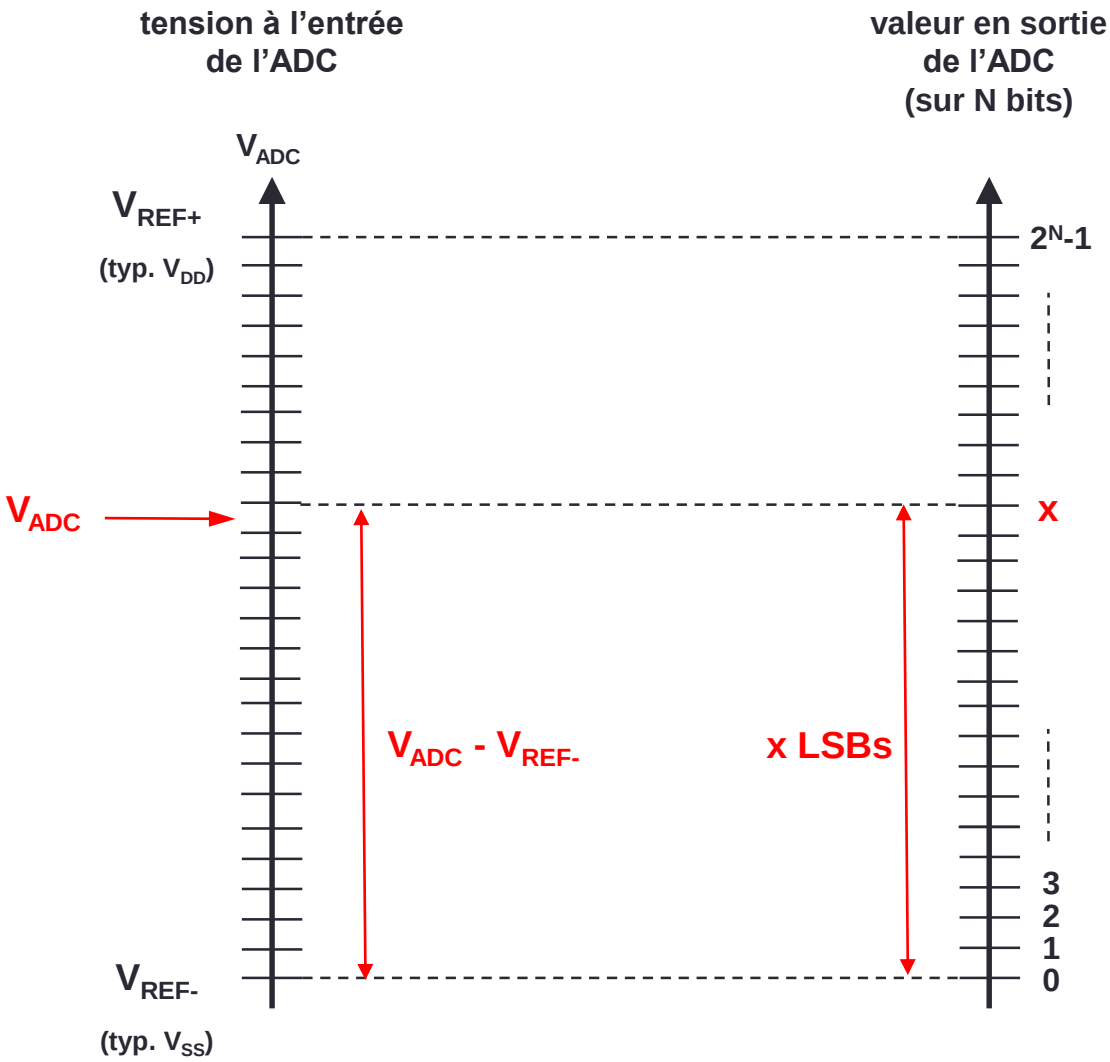
0110…110

0011…011

0001…010

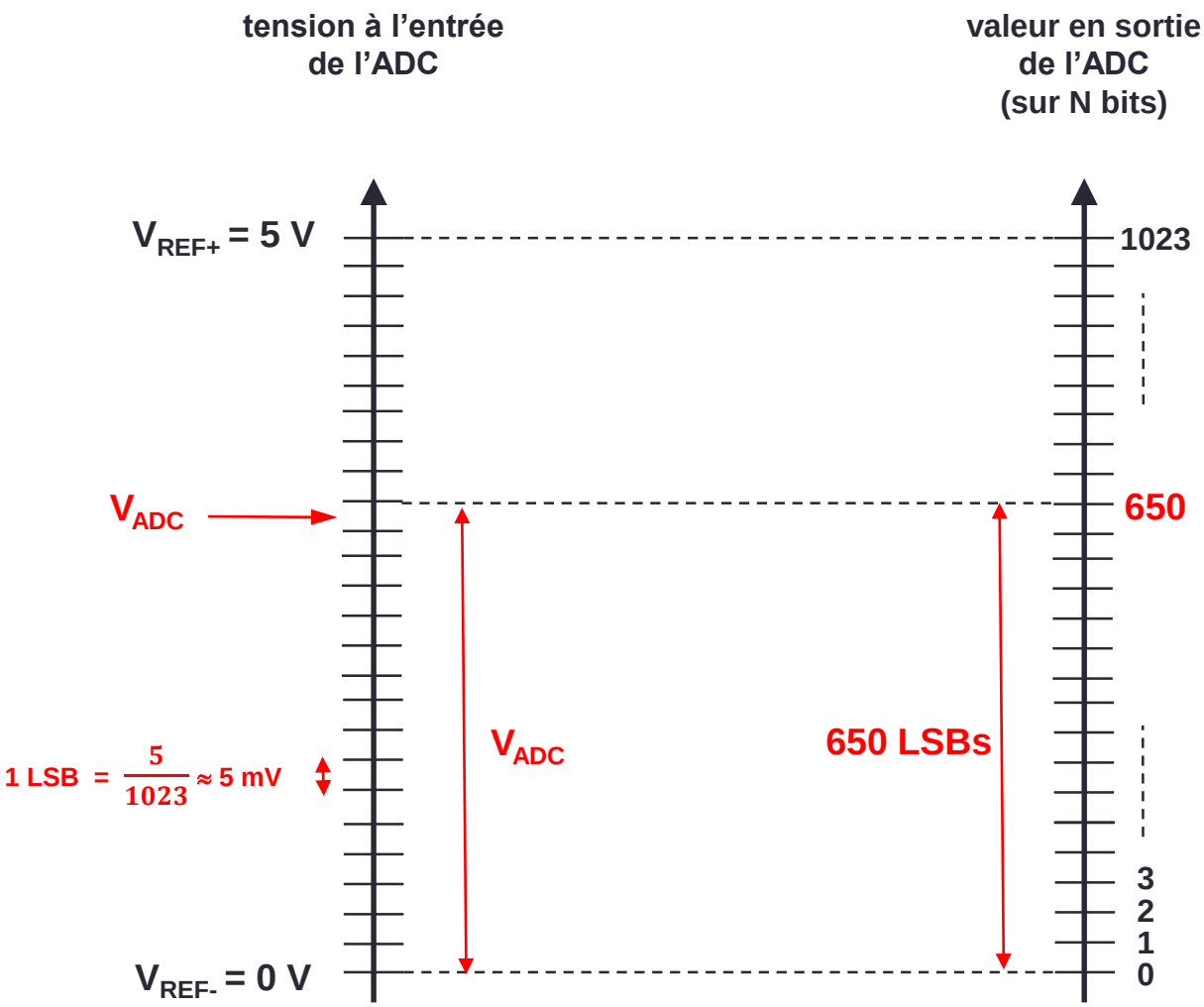

tension à l'entrée de l'ADC
valeur en sortie de l'ADC (sur N bits)
V_{ADC}
V_{REF+}
(typ. V_{DD})
V_{REF-}
(typ. V_{SS})
1111…1111
1111…1110
1111…1101
0000…0011
0000…0010
0000…0001
0000…0000
dynamique d'entrée
résolution =
1 intervalle:
$\frac{(V_{REF+} - V_{REF-})}{2^N - 1}$
(appelé LSB)
2^N valeurs possibles
(2^N-1 intervalles)

En lisant la valeur disponible en sortie de l'ADC, on connaît la tension présente sur son entrée.



$$(V_{ADC} - V_{REF-}) = x \times \frac{(V_{REF+} - V_{REF-})}{2^N - 1}$$

Exemple: convertisseur 10 bits, V_{REF-} = 0 V, V_{REF+} = 5 V



V_{ADC}

=

$650 \times \frac{5}{1023}$

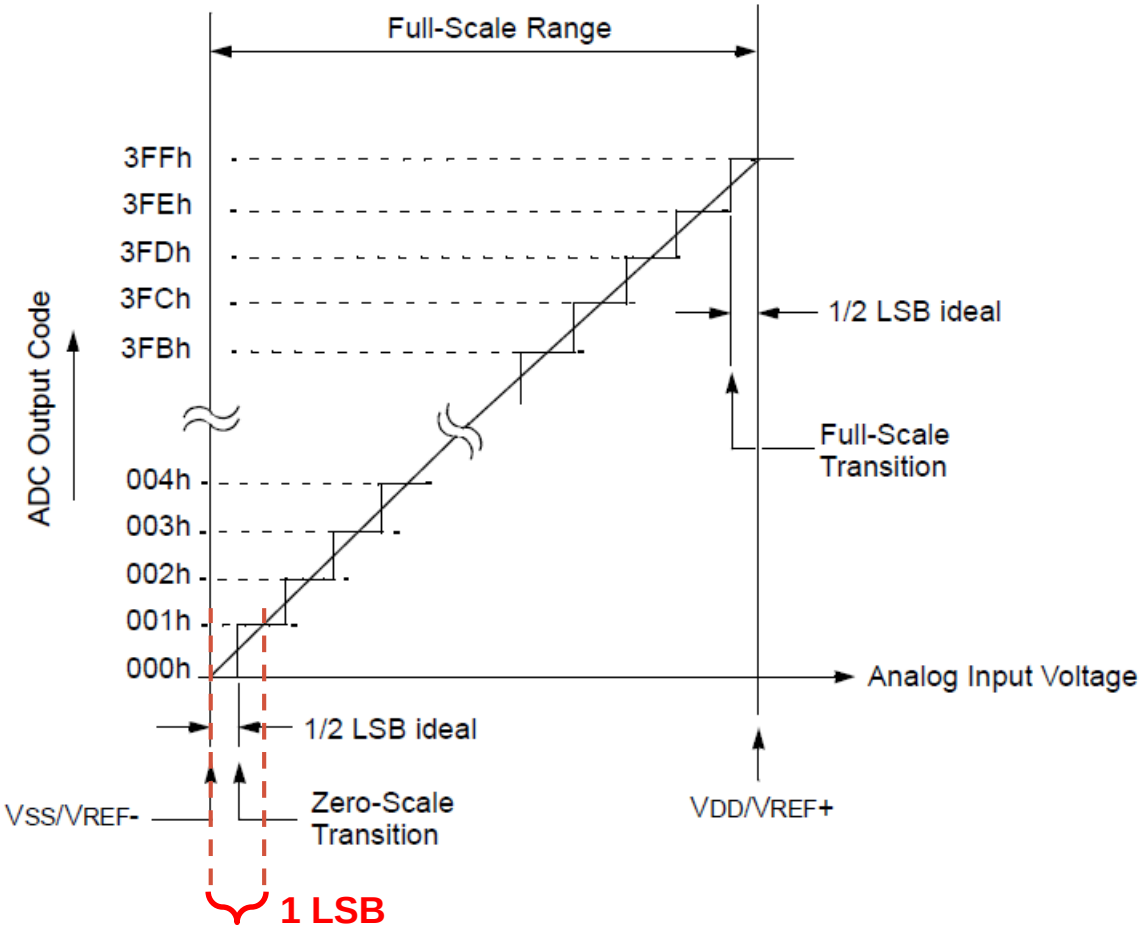
≈

3,177 V

(à 5 mV près)

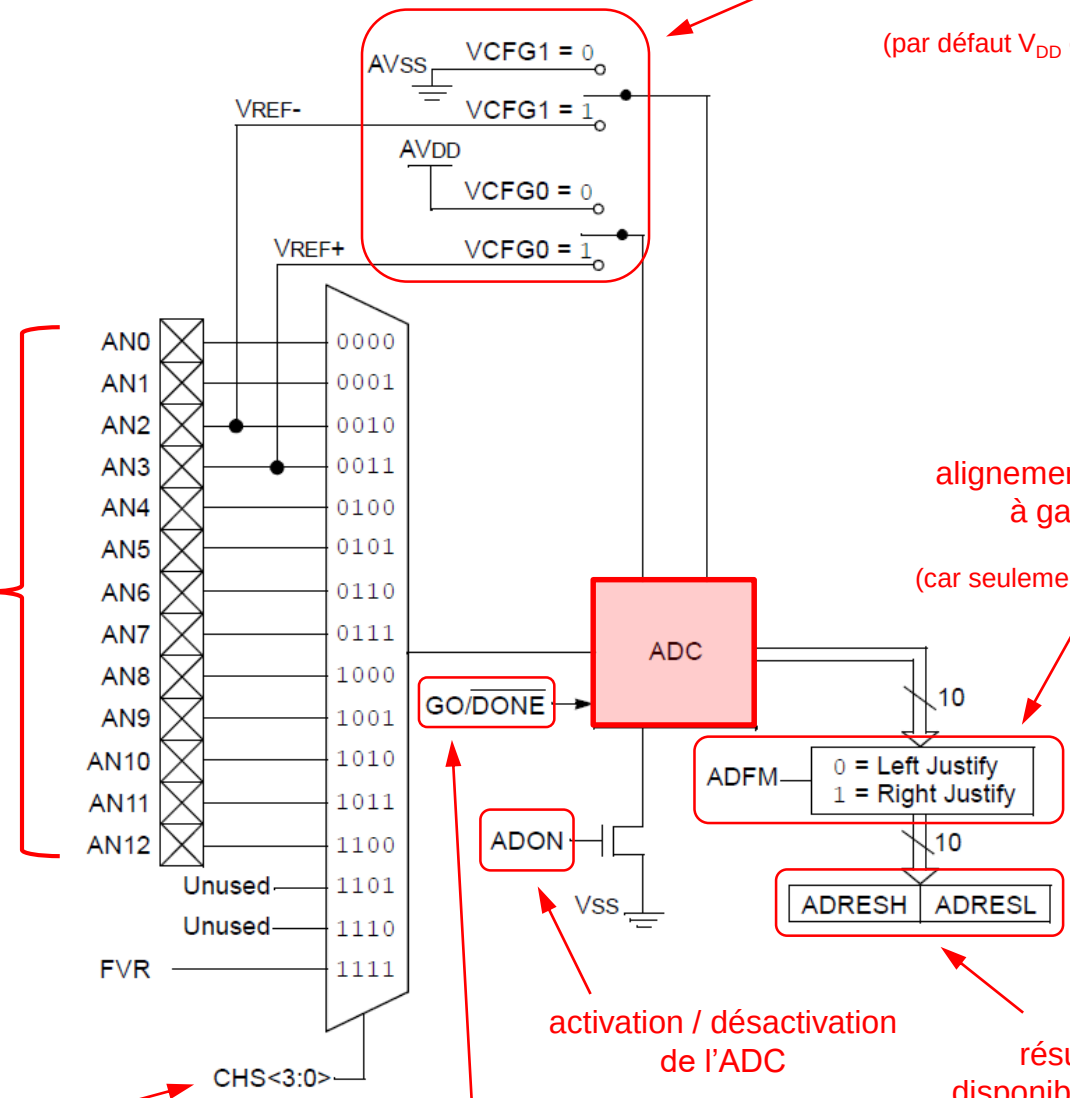
Exemple du PIC18F45K20: 1 ADC 10 bits disponible dans le microcontrôleur

ADC TRANSFER FUNCTION



choix possible de V_{REF+} et V_{REF-}

(par défaut V_{DD} et V_{SS})



13 entrées analogiques possibles

(mais un seul ADC !)

alignement possible du résultat à gauche ou a droite

(car seulement 10 bits utilisés sur les 16)

choix de l'entrée

lancement d'une conversion + flag de fin de conversion

activation / désactivation de l'ADC

résultat de la conversion disponible dans 2 registres dédiés

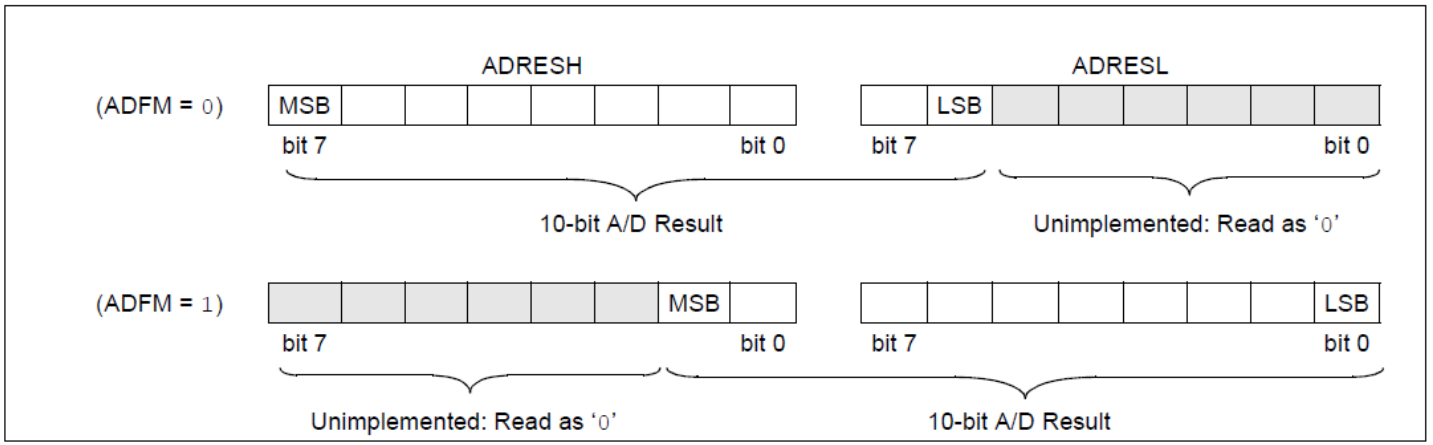
(10 bits utilisés sur les 16)

Le résultat de la conversion (sur 10 bits) peut être

- aligné à droite (plus pratique si on travaille sur 10 bits),
- aligné à gauche (plus pratique si on travaille sur 8 bits).

configurable par bit dédié (ADFM) dans registre dédié (ADCON2)

10-BIT A/D CONVERSION RESULT FORMAT



Principales étapes pour réaliser une conversion analogique-numérique:

1. configurer la (les) patte(s) concernée(s) en **entrée**(s) **analogique**(s) (registres TRIS et ANSEL),
2. configurer les tensions $\mathbf{V_{REF+}}$ et $\mathbf{V_{REF-}}$ (bits VCFG[1..0], registre ADCON1),
   ainsi que la **fréquence** de fonctionnement de l'ADC (bits ADCS[2..0], registre ADCON2),
3. configurer **l'alignement** du résultat (bit ADFM, registre ADCON2),
4. **sélectionner** l'entrée à convertir (bits CHS[3..0], registre ADCON0),
5. **activer** l'ADC (bit ADON, registre ADCON0),
6. **démarrer** une conversion (bit GO/$\overline{\text{DONE}}$, registre ADCON0: à mettre à 1 par software),
7. **attendre** la fin de la conversion (bit GO/$\overline{\text{DONE}}$, registre ADCON0: remis à 0 par hardware),
8. **lire** le résultat (registres ADRESH et ADRESL).

REGISTER 19-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

REGISTER 19-2: ADCON1: A/D CONTROL REGISTER 1

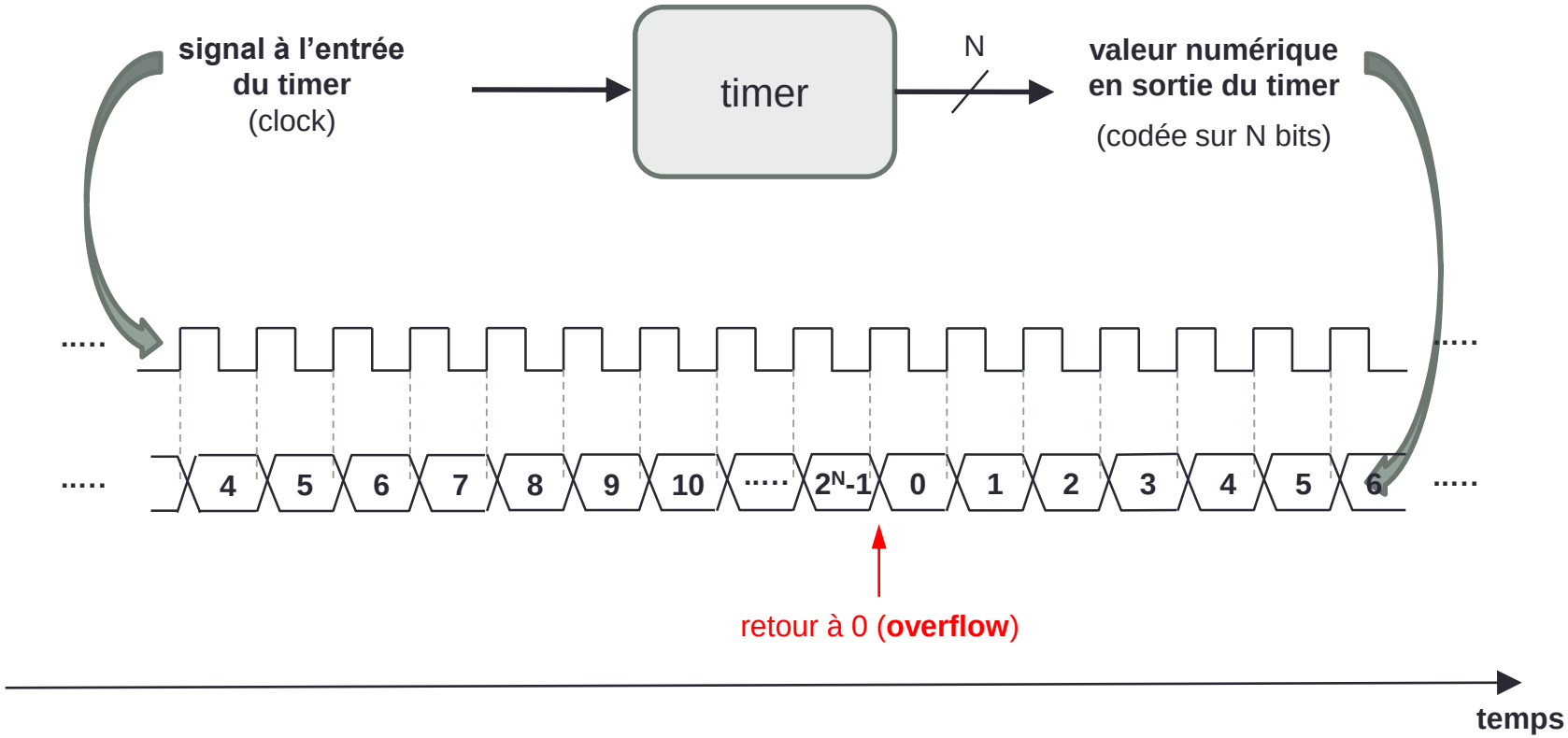
U-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	—	VCFG1	VCFG0	—	—	—	—
bit 7							bit 0

REGISTER 19-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

4. Les timers

Un timer est un compteur: il compte les fronts appliqués sur son entrée (clock) et fournit ce résultat en sortie, sur N bits.



4.1 Application des timers

- **Comptage** d'événements extérieurs
  si entrée du timer = une patte d'entrée du microcontrôleur

- **Temporisation**
  si entrée du timer = horloge interne ⇒ « **overflow** » à intervalle régulier
  (cet événement peut éventuellement déclencher une interruption)

- **Mesure d'intervalle de temps**
  si entrée du timer = patte d'entrée du microcontrôleur et fonction « **Input Capture** » activée
  (la valeur du timer est automatiquement sauvegardée lors d'un changement d'état de l'entrée: la différence du timer entre 2 changements successifs indique le temps écoulé)

- **Déclenchement d'une action** sur une valeur particulière du timer
  si fonction « **Output Compare** » activée
  (une patte de sortie change d'état lorsque la valeur du timer atteint une valeur particulière)

- Génération de signal **PWM** (Pulse Width Modulation)

- **Sécurité** de fonctionnement
  si fonctionnalité « **Watchdog** » activée
  (lorsque le timer particulier « Watchdog » atteint sa valeur limite, il provoque un RESET du microcontrôleur)

4.2 Exemple du PIC18F45K20

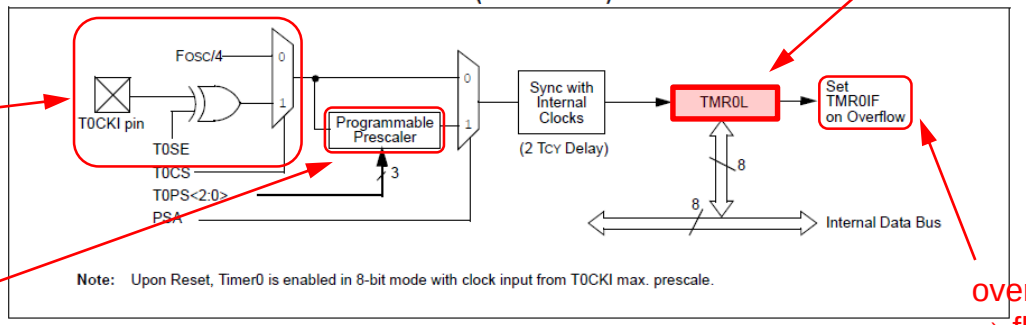
1 timer 8 ou 16 bits (Timer0)
2 timers 16 bits (Timer1 et Timer3)
1 timer 8 bits (Timer2)

Exemple du Timer0:

timer 8 bits)
(accessible en lecture et écriture)

choix entre
- horloge interne (timer)
- horloge externe (compteur d'événements)

FIGURE 12-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)

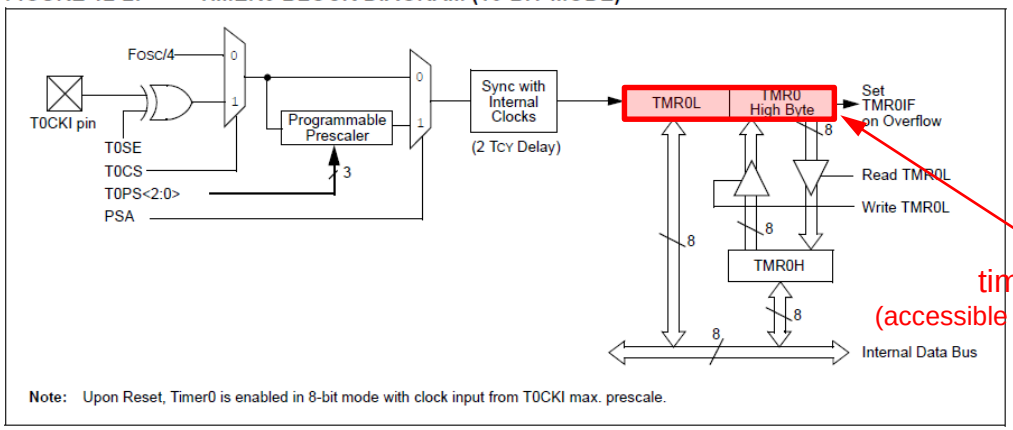


Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

ralentissement possible
de la fréquence d'horloge
($^1/_2$ à $^1/_{256}$)

overflow
⇒ flag mis à 1
(+ interruption possible)

FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

timer 16 bits)
(accessible en lecture et écriture)

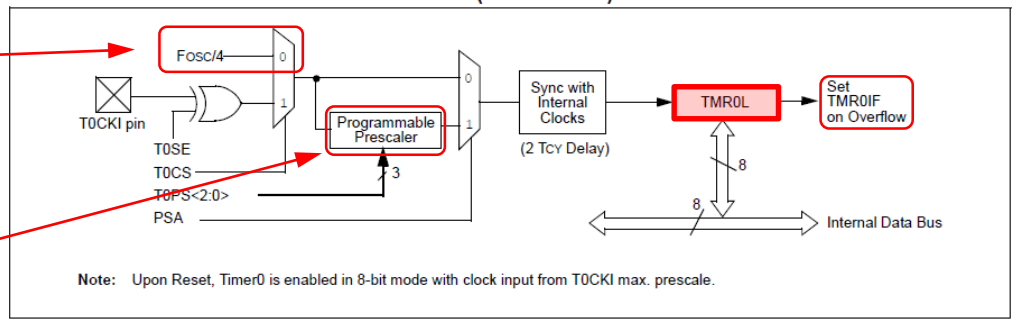
Exemple du Timer0 (suite):

Mode 8 bits

cadencement par horloge interne
avec f_{osc} = 16 MHz
⇒ f_{TMR} = 4 MHz

prescaler non utilisé

FIGURE 12-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)



timer cadencé à $f_{osc}/4$ = 4 MHz (1 incrément toutes les ¼ µs)

1 cycle complet : 256 périodes d'horloge (car timer 8 bits) ⇒ 1 overflow toutes les 256 × ¼ µs = 64 µs

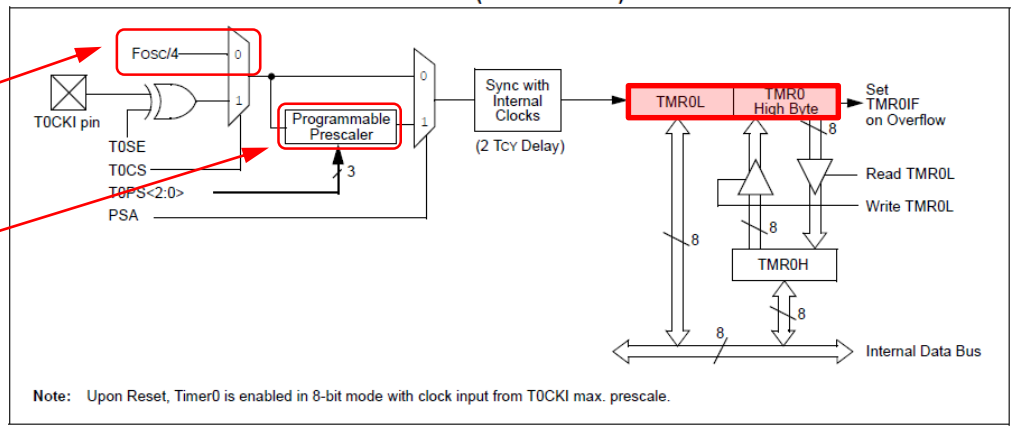
⇒ **1 interruption possible toutes les 64 µs**

Mode 16 bits

cadencement par horloge interne
avec f_{osc} = 16 MHz
⇒ f_{TMR} = 4 MHz

prescaler 1/256 utilisé

FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



timer cadencé à $f_{osc}/4 \div 256$ = 15 625 Hz (1 incrément toutes les 64 µs)

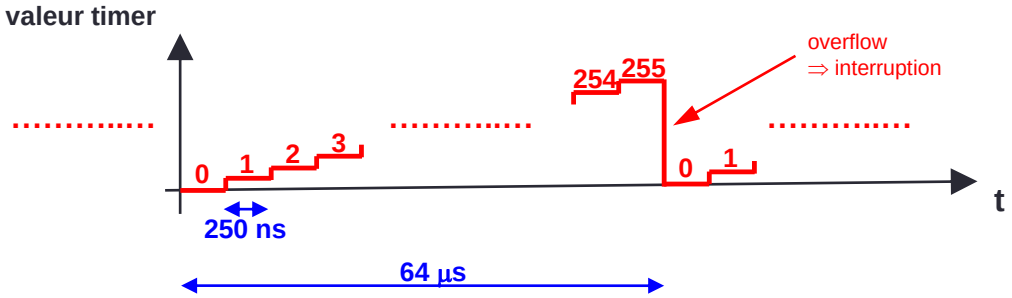
1 cycle complet : 65536 périodes d'horloge (car timer 16 bits) ⇒ 1 overflow toutes les 65536 × 64 µs = 4,194… s

⇒ **1 interruption possible toutes les 4,2 secondes (environ)**

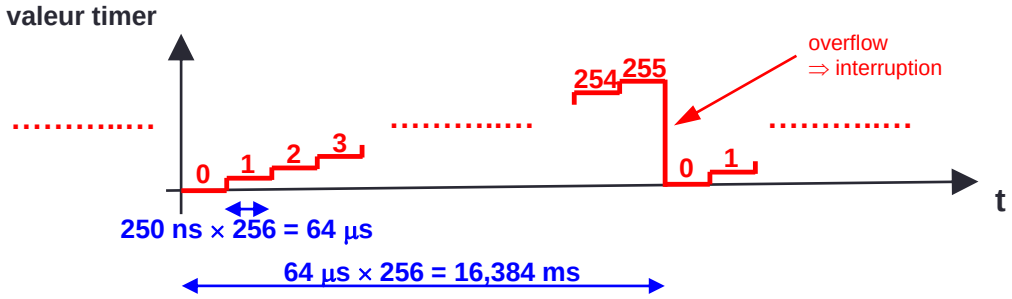
4.3 Comment générer une interruption toutes les secondes (exactement) avec le Timer0 ?

Hypothèse de départ: utilisation de l'horloge interne, avec $\mathbf{f_{osc} = 16\ MHz}$
⇒ horloge du timer $\mathbf{f_{TMR} = 4\ MHz}$ (**période = 250 ns**)

- **Mode 8 bits sans prescaler** (1 cycle complet = 2^8 = 256 incrémentations)
  ⇒ 1 interruption possible toutes les **64 µs** ⇒ **insuffisant**

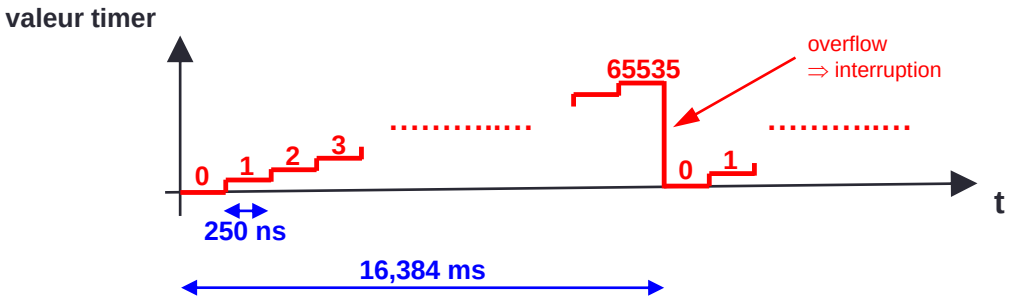


- **Mode 8 bits avec prescaler maximal** (1/256)
  ⇒ 1 interruption possible toutes les 256 × 64 µs = **16,384 ms** ⇒ **encore insuffisant**



⇒ **Mode 16 bits indispensable**

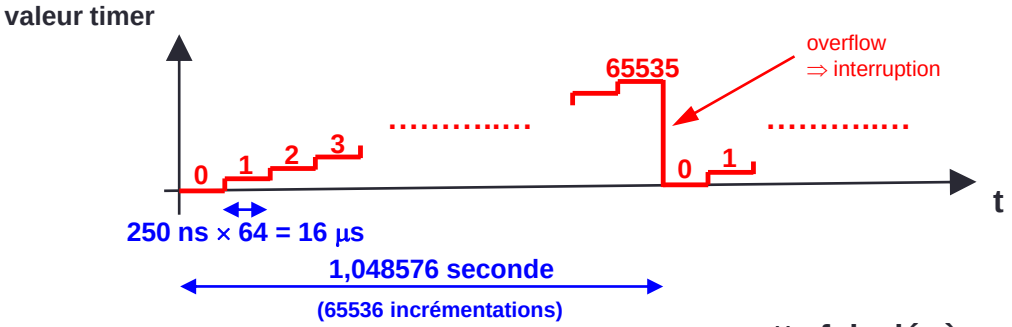
- **Mode 16 bits sans prescaler** (1 cycle complet = 2^{16} = 65536 incrémentations)



1 seconde = 16,384ms × 61,03516

⇒ **toujours ≈ 61 fois trop rapide**
⇒ **il faut ralentir le timer (prescaler possible: 64)**

- **Mode 16 bits avec prescaler 1/64**



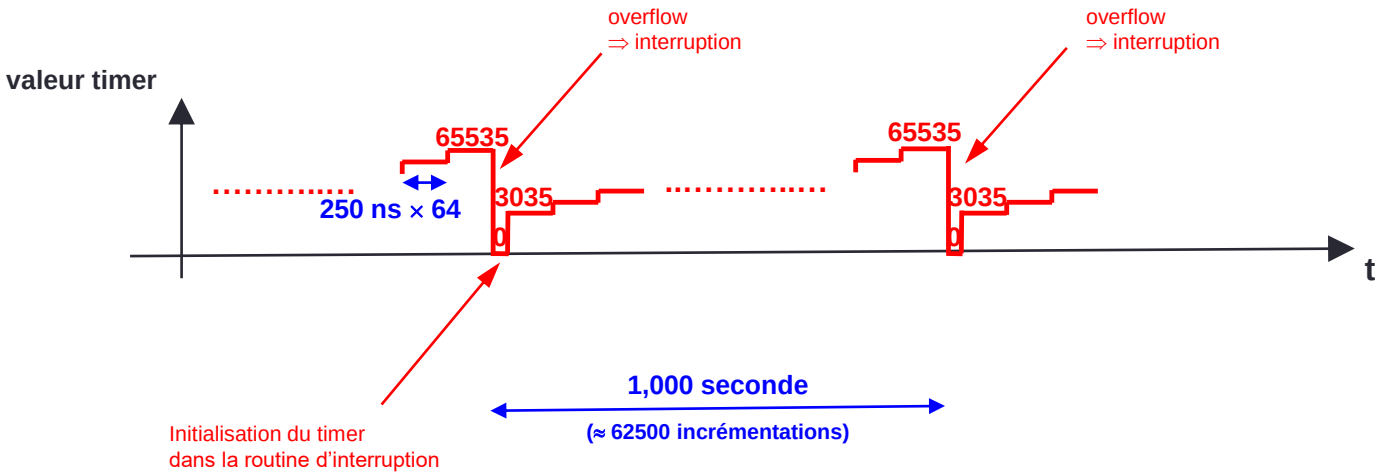
⇒ **cette fois, légèrement trop lent**
⇒ **on va ajuster le nombre d'incrémentations entre 2 overflows**

Solution d'ajustement: initialiser le timer à une valeur différente de 0

le timer s'incrémente toutes les 250 ns × 64 = 16 µs
⇒ 1 seconde correspond à 62500 incrémentations

l'overflow se produit lorsque le timer a atteint sa valeur maximale (65535)

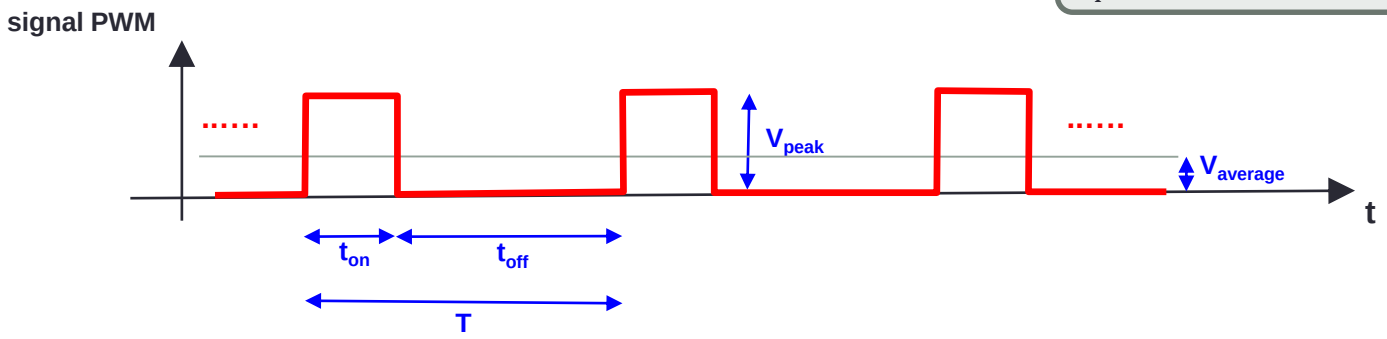
⇒ en chargeant le timer à la valeur **(65535 – 62500) = 3035**, le prochain overflow arrivera exactement 1 seconde plus tard
(à faire au début de la routine d'interruption)



4.4 Fonction PWM (Pulse Width Modulation)

Objectif: Générer un signal tout-ou-rien périodique,
d'amplitude fixe
de fréquence fixe,
de rapport cyclique ajustable.

$\frac{t_{on}}{T} = \theta$ (rapport cyclique)



Intérêt: La valeur moyenne du signal dépend du rapport cyclique.

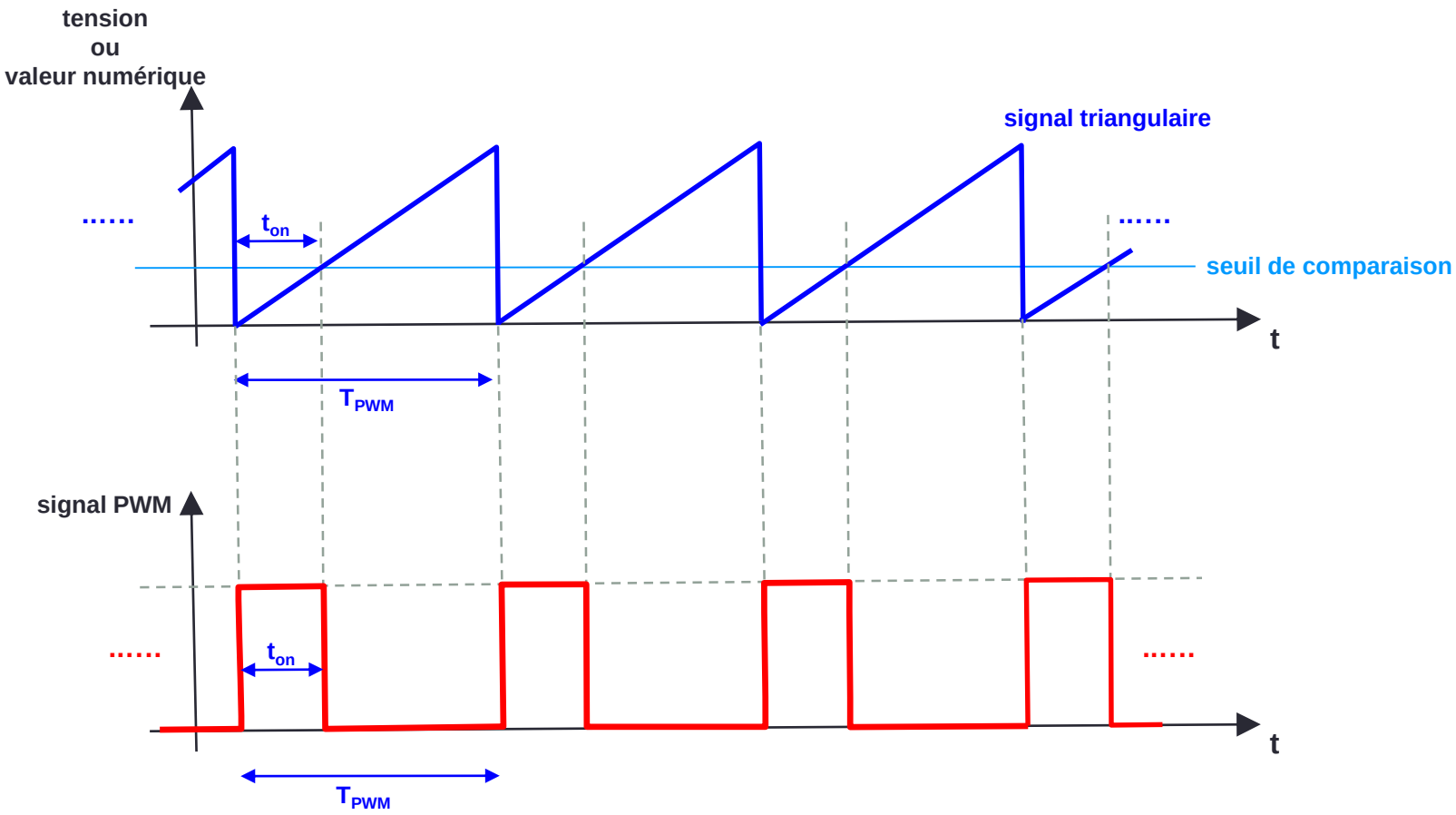
$V_{average} = V_{peak} \times \theta$

Après filtrage passe-bas (pour ne garder que la valeur moyenne), on a l'équivalent d'une conversion numérique-analogique.

Si la fréquence est suffisamment « grande » (≈ kHz):

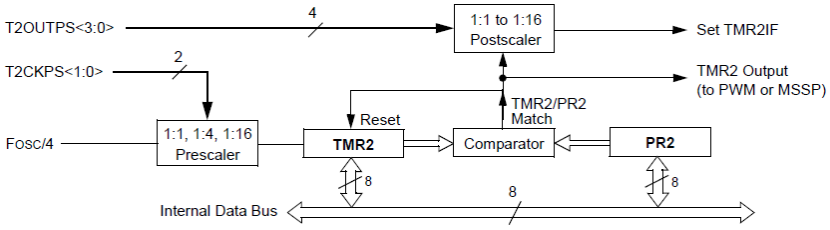
- appliqué aux bornes d'un moteur à courant continu, on obtient un variateur de vitesse,
- appliqué aux bornes d'une lampe ou LED, on obtient un variateur de lumière,
- idem pour chauffage, etc…

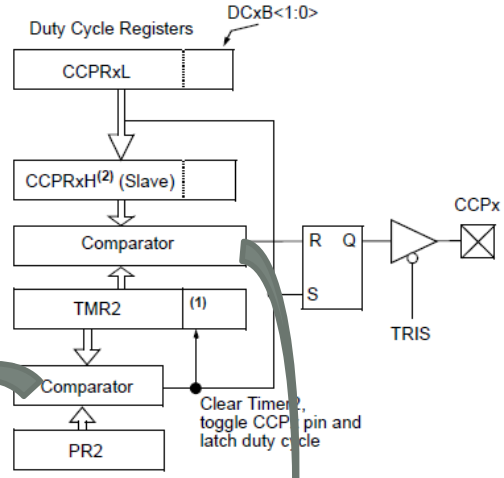
Principe de génération: comparer un signal de forme triangulaire avec un seuil de valeur fixe



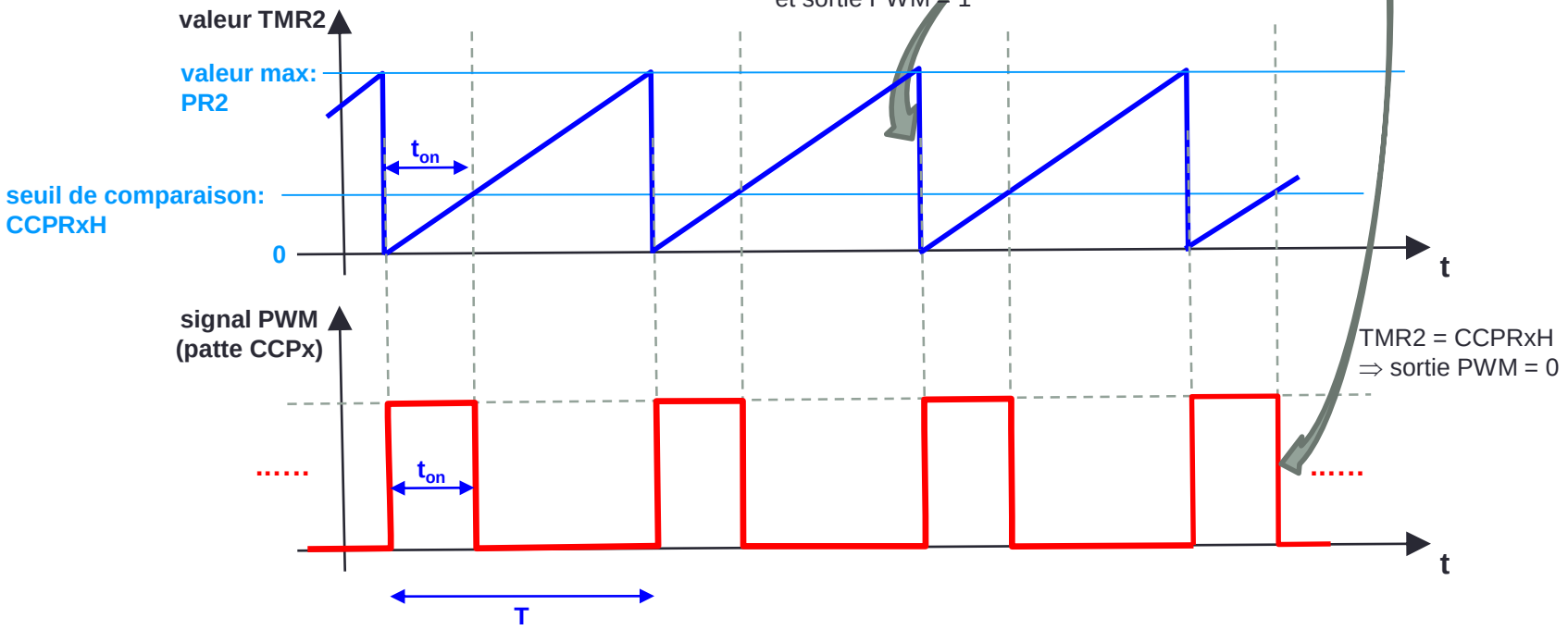
Dans un microcontrôleur: Le signal triangulaire est fourni par la sortie d'un timer.

Timer2 pour PIC18F45K20 (2 signaux PWM disponibles: x = 1 ou 2)





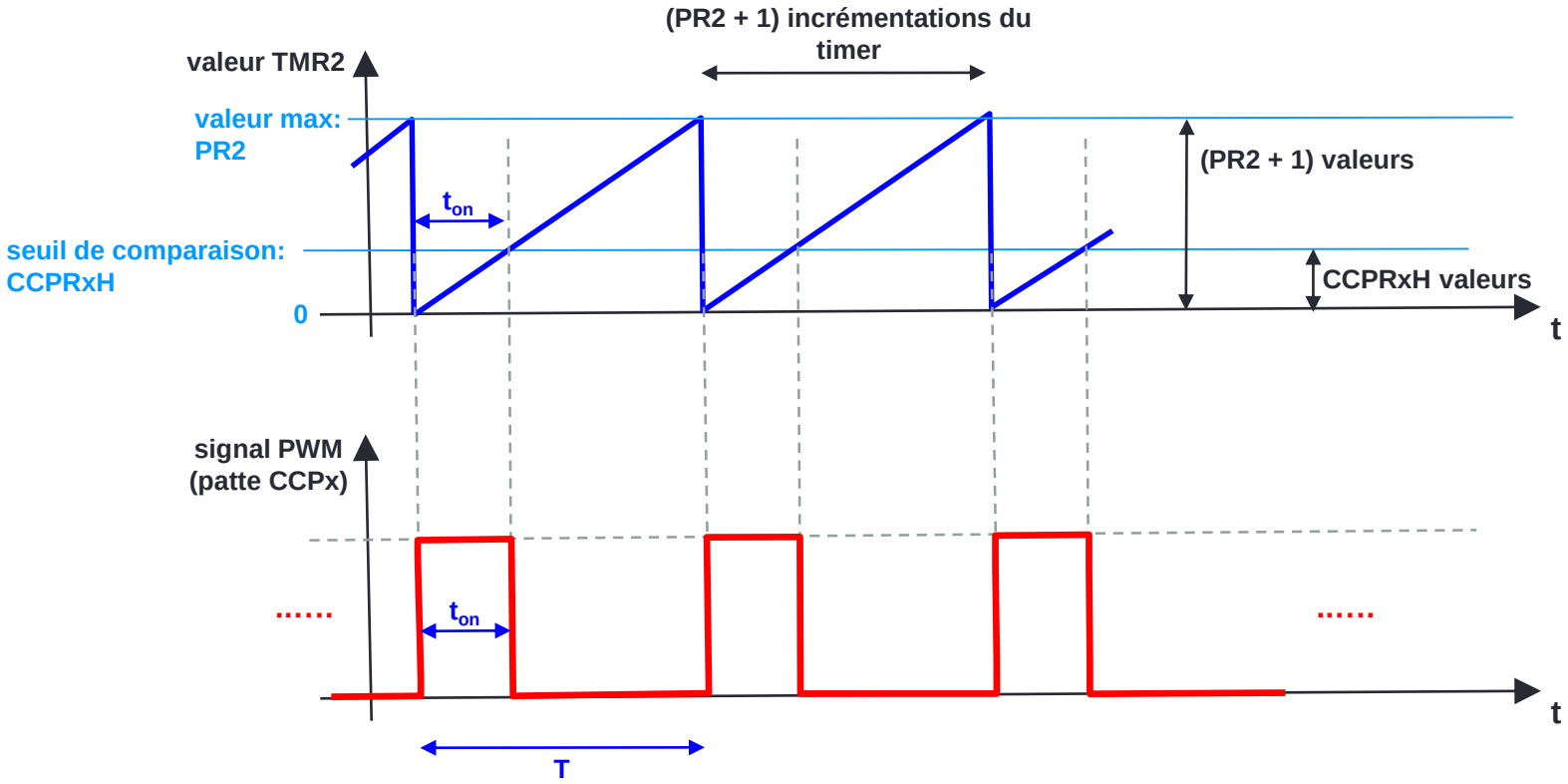
TMR2 = PR2
⇒ TMR2 = 0
et sortie PWM = 1



TMR2 = CCPRxH
⇒ sortie PWM = 0

La période du signal PWM est réglée grâce à PR2: $T_{PWM} = (PR2 + 1) \times T_{CLK\_TMR2}$

Le rapport cyclique est réglé grâce à CCPRxH: $\theta = {}^{CCPRxH} / {}_{(PR2 + 1)}$



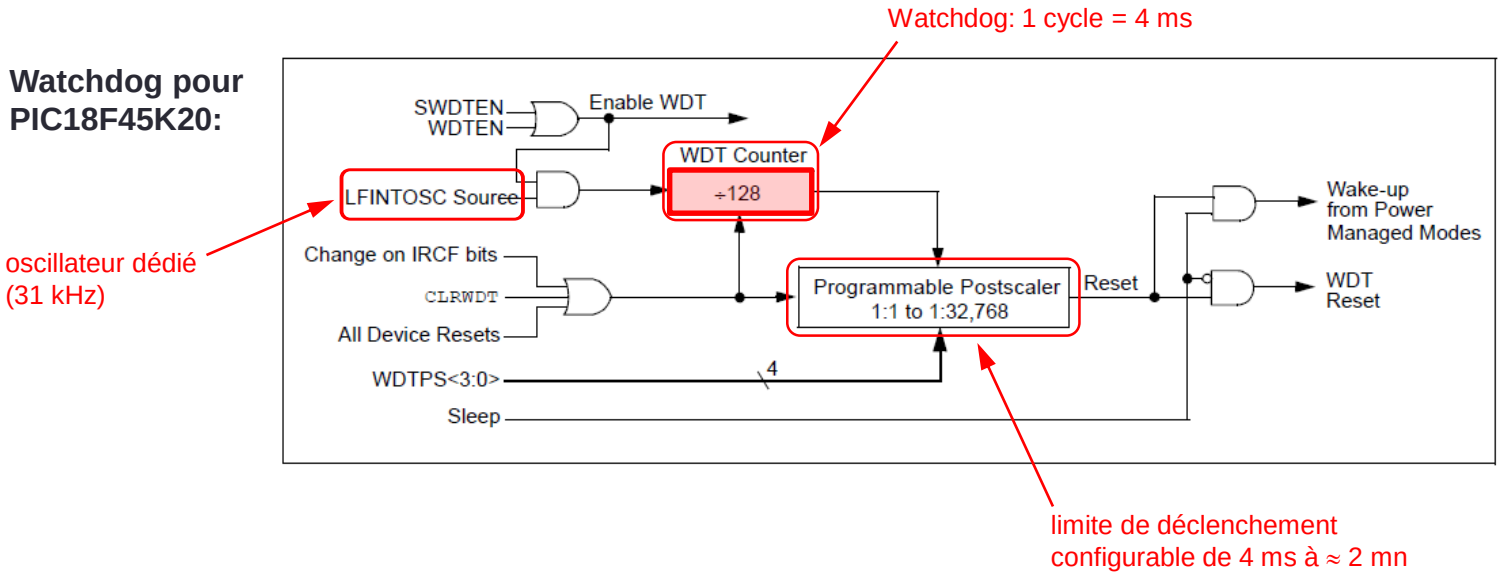
4.5 Fonction Watchdog (« chien de garde »)

Objectif: Détecter un dysfonctionnement durant l'exécution d'un programme, et déclencher dans ce cas un RESET de la CPU,

ou

réveiller le microcontrôleur au bout d'un temps connu.

Watchdog pour PIC18F45K20:



Mise en œuvre: Le programme doit remettre régulièrement à 0 le Watchdog (instruction **CLRWDT**), avant qu'il n'atteigne sa valeur limite.

Si, pour une raison quelconque, cette action ne se fait pas à temps, le Watchdog atteindra sa valeur limite et déclenchera un RESET.

5. Autres périphériques

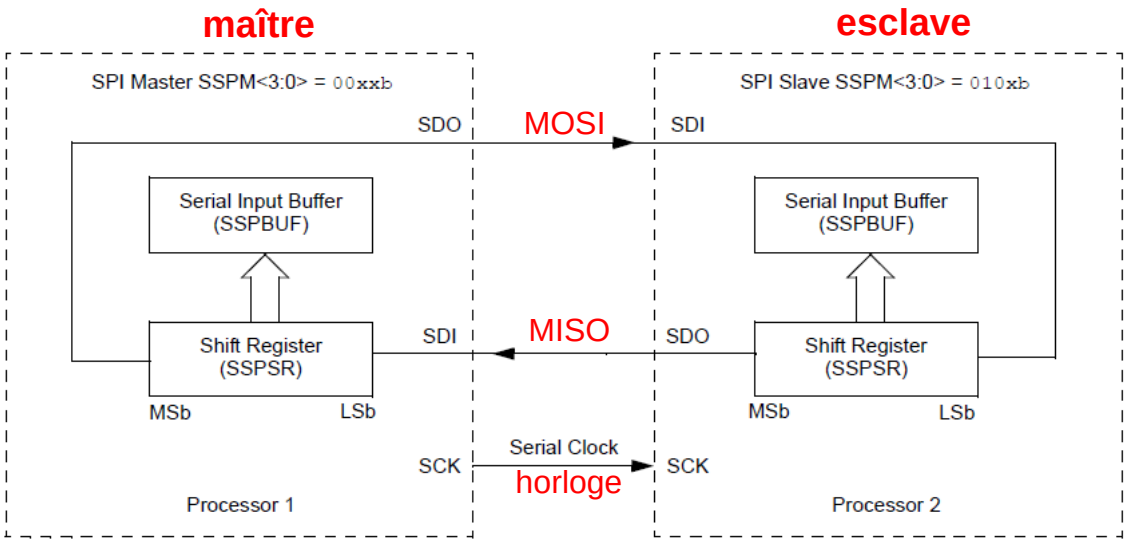
Liste non exhaustive:

- Liaison **SPI** (Serial Peripheral Interface)
  communication synchrone maître – esclaves full-duplex
- Liaison **I²C** (Inter Circuit Communication)
  communication synchrone maître – esclaves half-duplex
- Liaison **UART** (Universal Asynchronous Receiver Transmitter)
  communication asynchrone point-à-point full-duplex
- Liaison **CAN** (Controller Area Network)
  communication asynchrone multi-maîtres (très utilisé dans le milieu automobile)
- Liaison **USB** (Universal Serial Bus)
  communication série à usage général
- …

5.1 Liaison SPI

Liaison série (bits envoyés 1 par 1), full-duplex (transmission et réception simultanées), synchrone (avec signal d'horloge), entre un maître et un ou plusieurs esclaves.

- La transmission est toujours initiée par le maître.
- Le maître sélectionne un (et un seul) esclave avec lequel communiquer.
- Le maître et l'esclave s'échangent simultanément des informations, découpées en octets.
- Le maître fournit l'horloge pour synchroniser l'échange.



MOSI: ***M**aster **O**ut **S**lave **I**n*

MISO: ***M**aster **I**n **S**lave **O**ut*