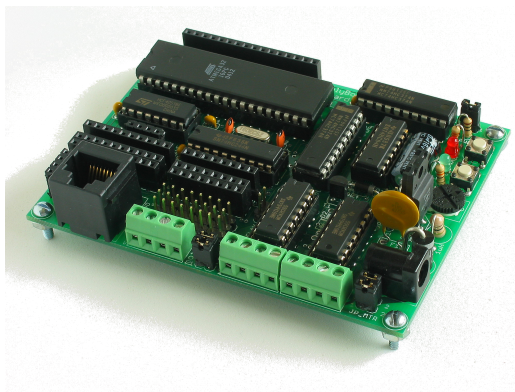
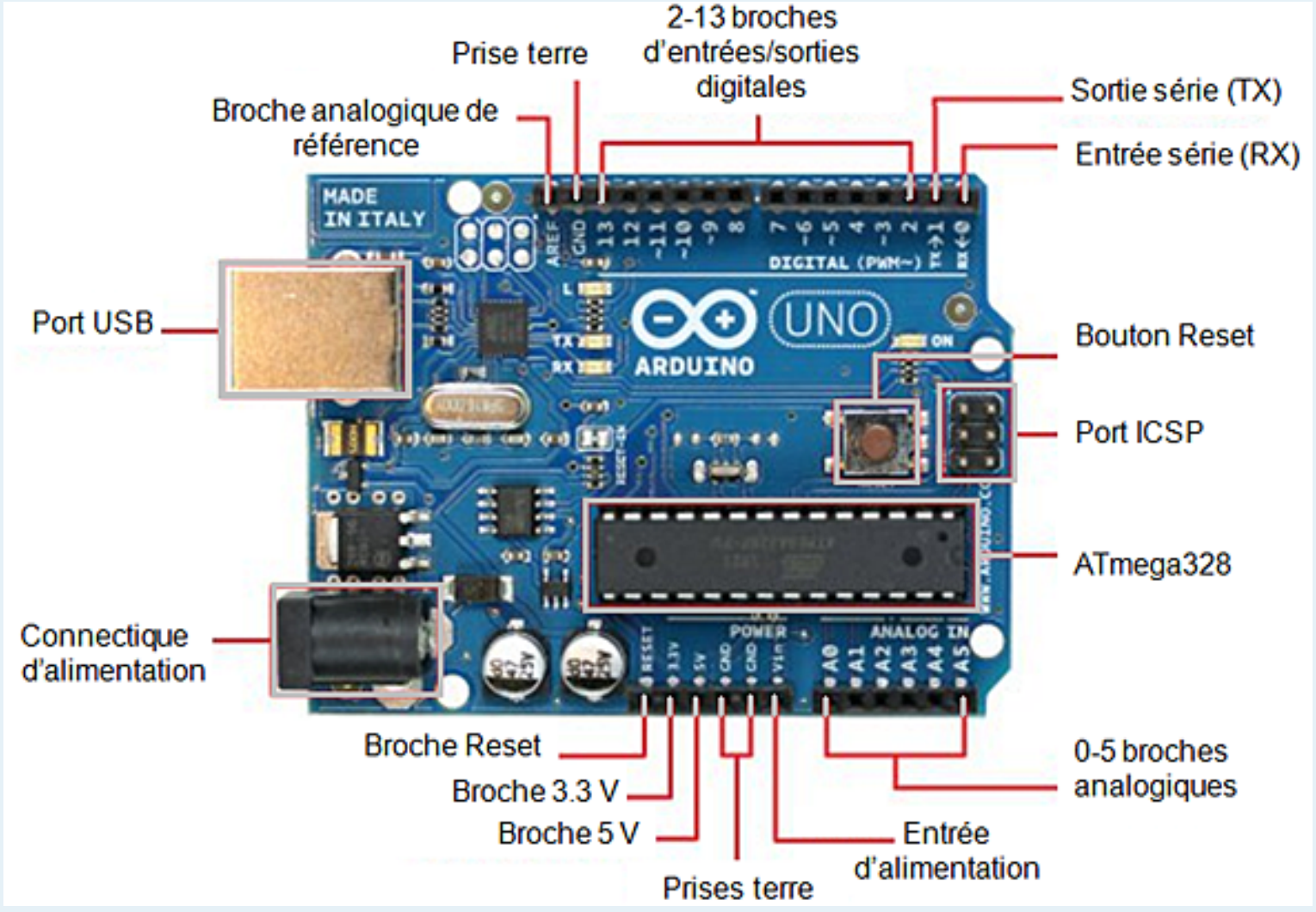
Microcontrôleur

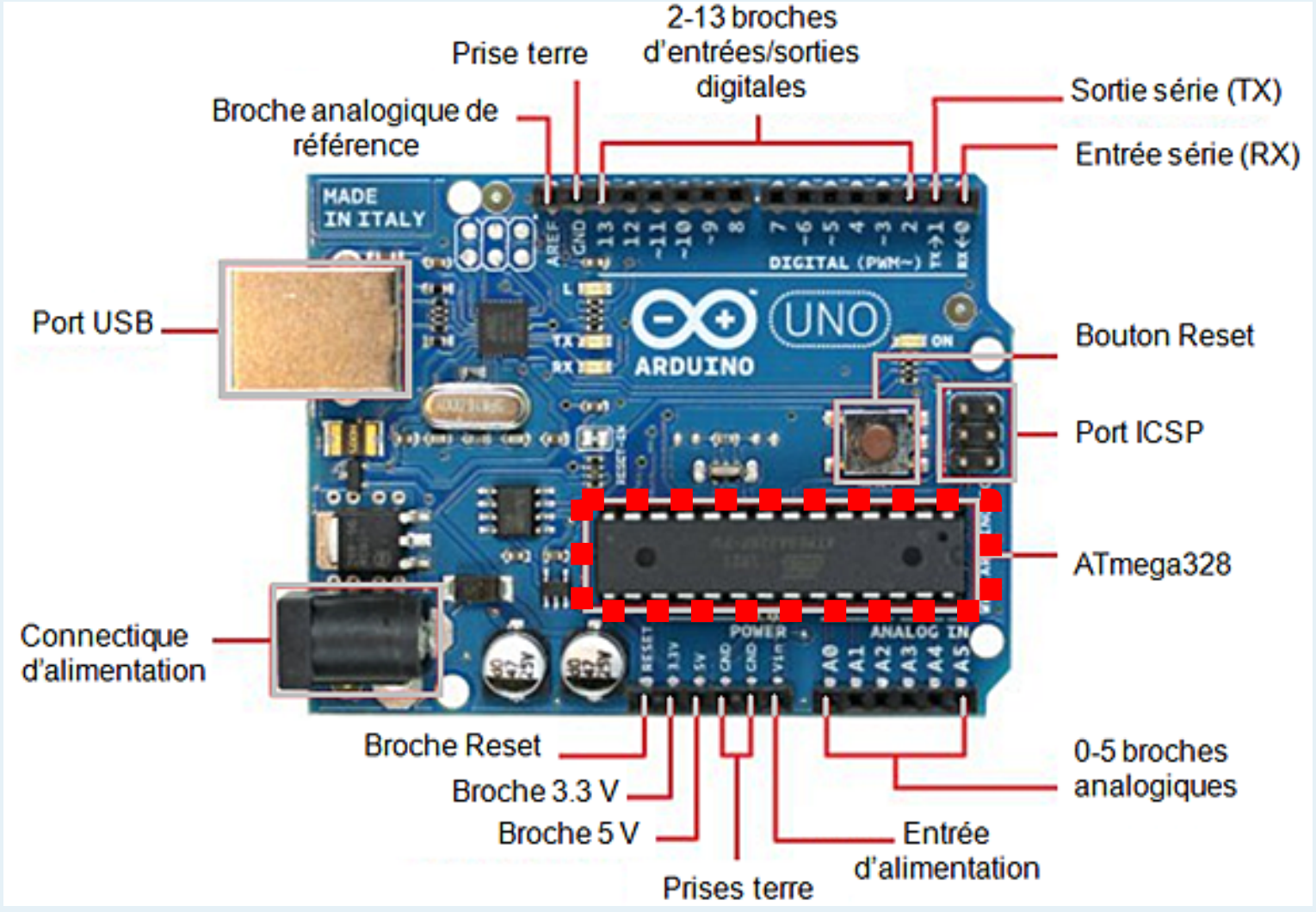
Utilisation des entrées/sorties numériques

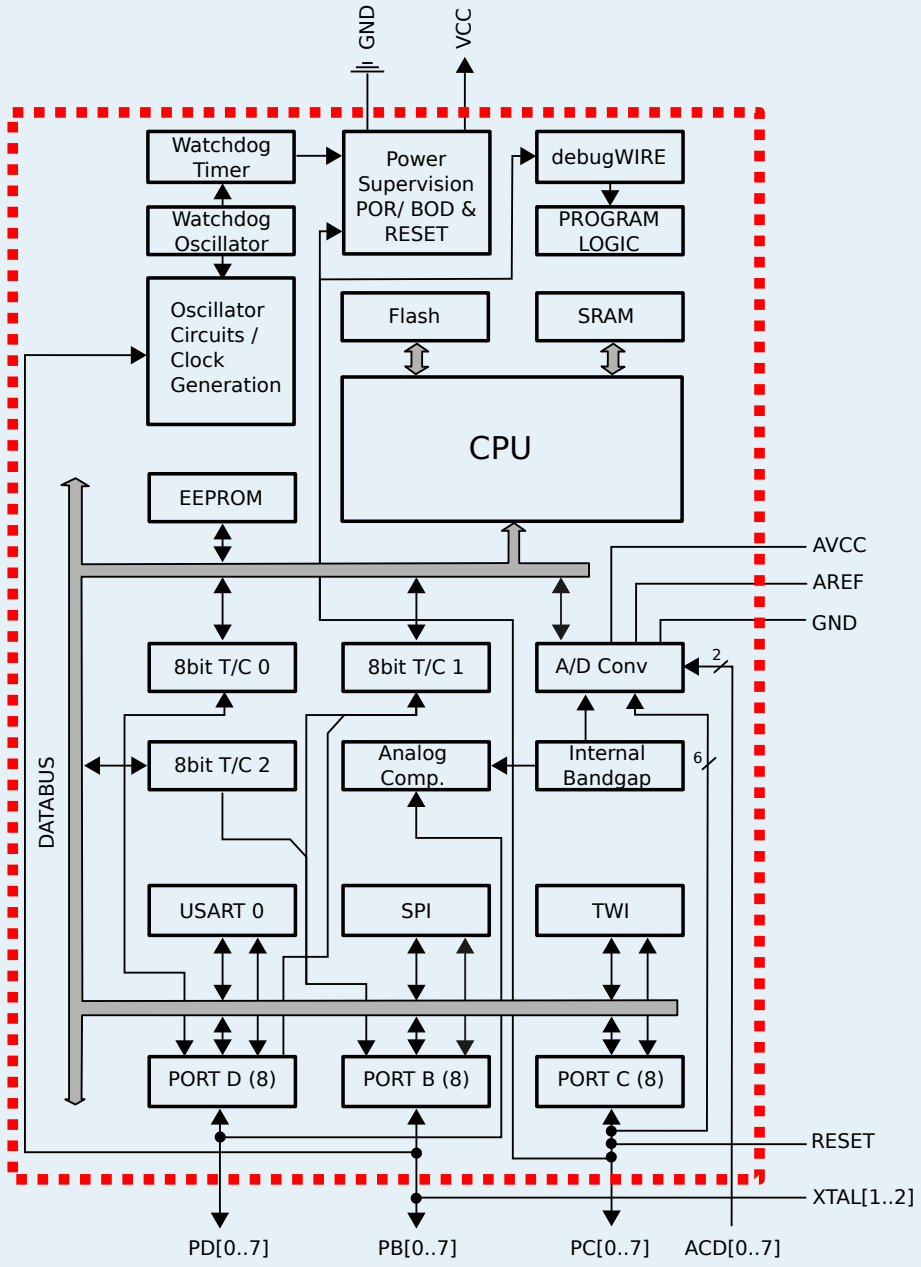
Nicolas Delanoue

Université d'Angers - Polytech Angers

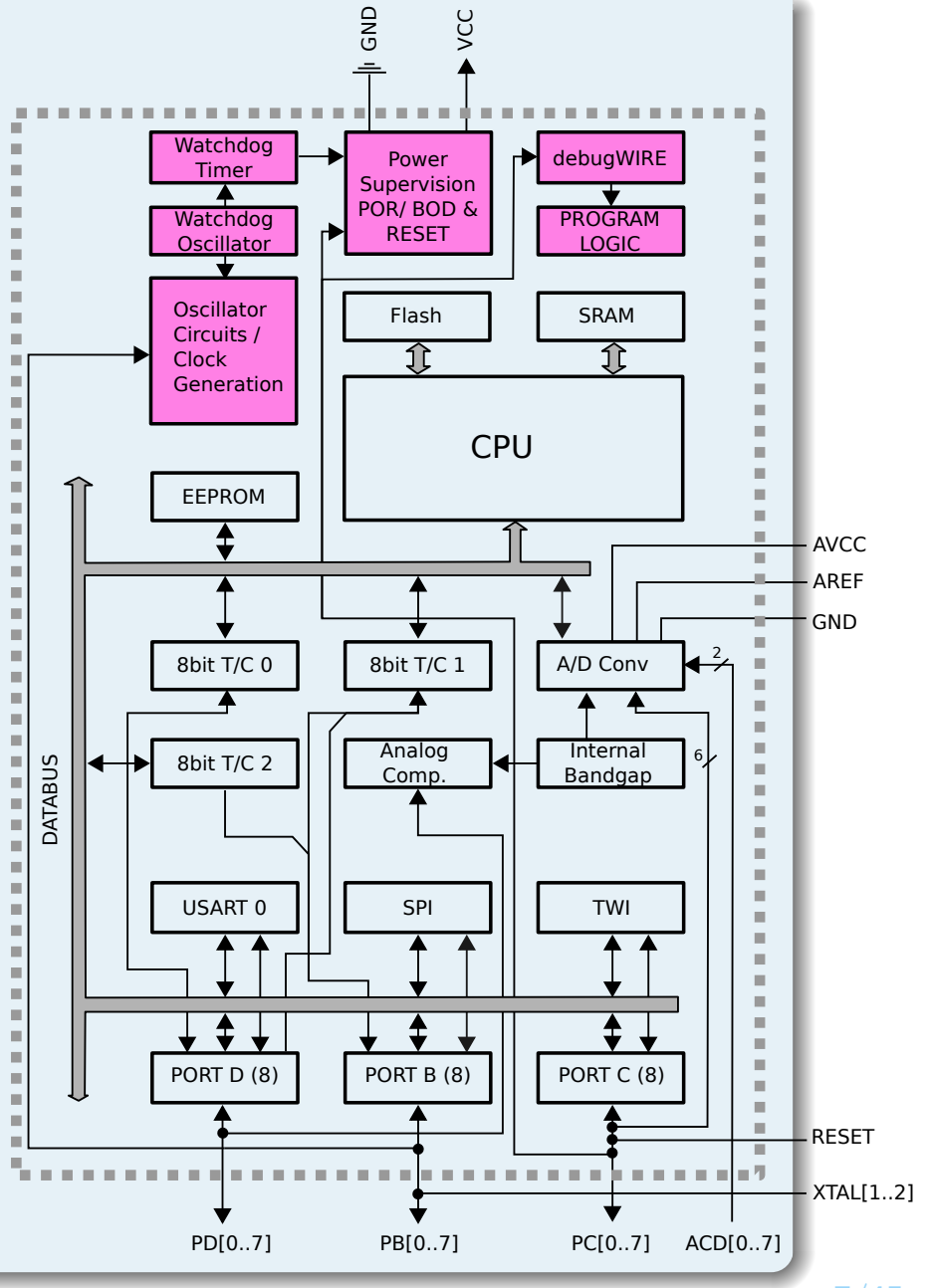



Prise terre
2-13 broches d'entrées/sorties digitales
Broche analogique de référence
Sortie série (TX)
Entrée série (RX)
Port USB
Bouton Reset
Port ICSP
ATmega328
Connectique d'alimentation
Broche Reset
Broche 3.3 V
Broche 5 V
Prises terre
Entrée d'alimentation
0-5 broches analogiques
MADE IN ITALY
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN

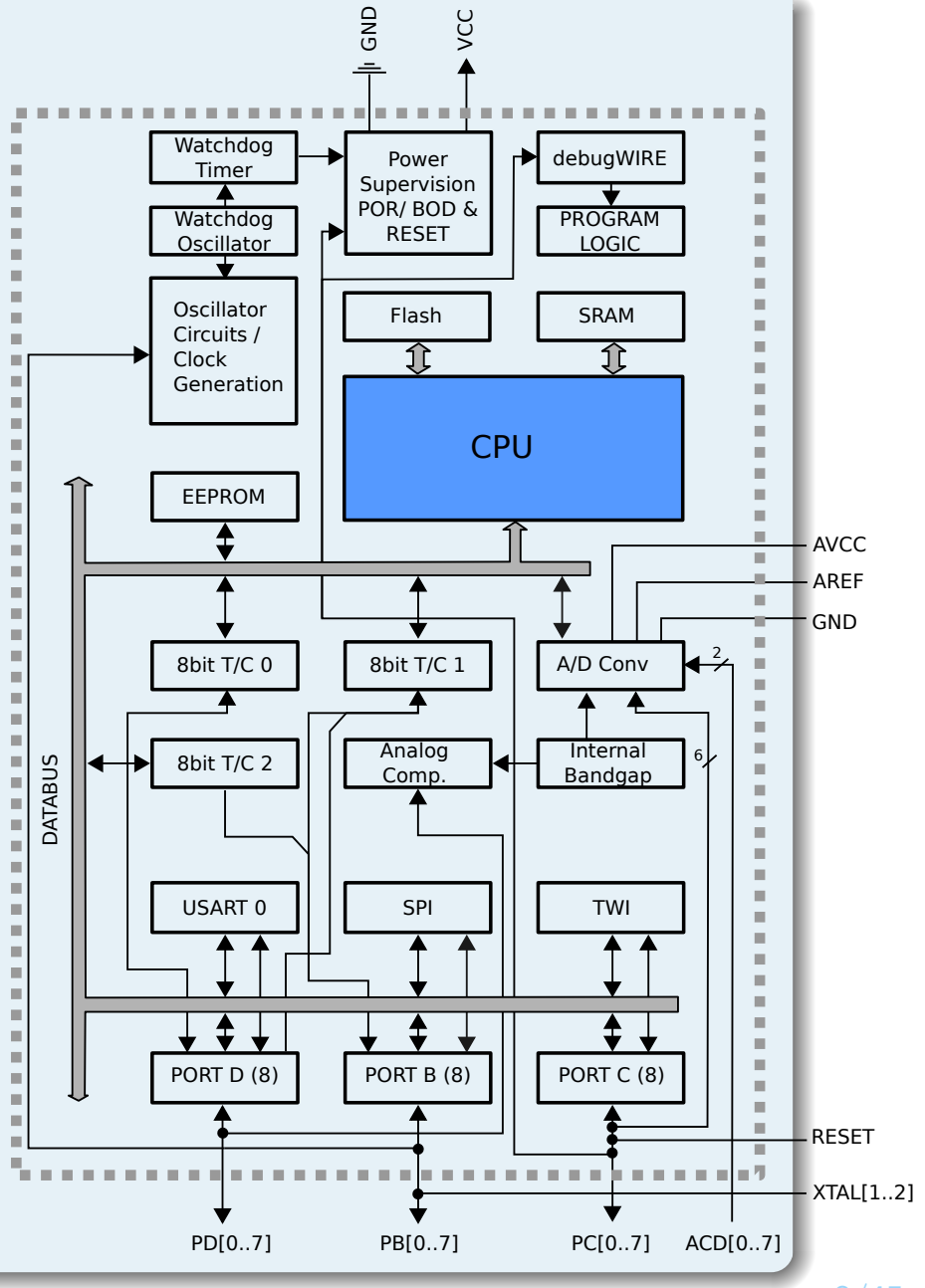

Prise terre
2-13 broches d'entrées/sorties digitales
Broche analogique de référence
Sortie série (TX)
Entrée série (RX)
Port USB
Bouton Reset
Port ICSP
ATmega328
Connectique d'alimentation
Broche Reset
Broche 3.3 V
Broche 5 V
Prises terre
Entrée d'alimentation
0-5 broches analogiques
MADE IN ITALY
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN


GND
VCC
Watchdog Timer
Watchdog Oscillator
Power Supervision POR/ BOD & RESET
debugWIRE
PROGRAM LOGIC
Oscillator Circuits / Clock Generation
Flash
SRAM
CPU
EEPROM
AVCC
AREF
GND
8bit T/C 0
8bit T/C 1
A/D Conv
2
8bit T/C 2
Analog Comp.
Internal Bandgap
6
DATABUS
USART 0
SPI
TWI
PORT D (8)
PORT B (8)
PORT C (8)
RESET
XTAL[1..2]
PD[0..7]
PB[0..7]
PC[0..7]
ACD[0..7]

- Horloges et alimentation
  - Au-delà de la portée de ce cours



- Microprocesseur
  - Vu lors du prochain cours



Comment stocker de l'information ?

FIGURE – Un bit peut être stocké à l'aide d'une bascule.

Comment stocker de l'information ?

FIGURE – Un bit peut être stocké à l'aide d'une bascule.

Un codage

Etat qui code 0

Etat qui code 1

Exemple

L'octet `00111110` peut être stocké via :

Exemple

L'octet `00111110` peut être stocké via :

Exemple

L'octet `00111110` peut être stocké via :


Exemple

L'octet `00111110` peut être stocké via :



Exemple

L'octet `00111110` peut être stocké via :

Exemple

L'octet `00111110` peut être stocké via :

Exemple

L'octet `00111110` peut être stocké via :

Exemple

L'octet `00111110` peut être stocké via :

Exemple

L'octet `00111110` peut être stocké via :

Bascule

Electroniquement,

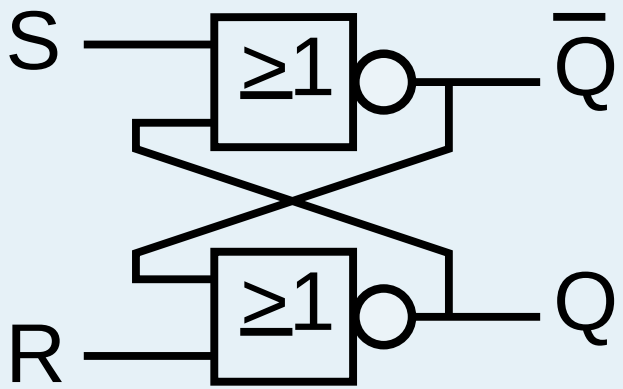


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

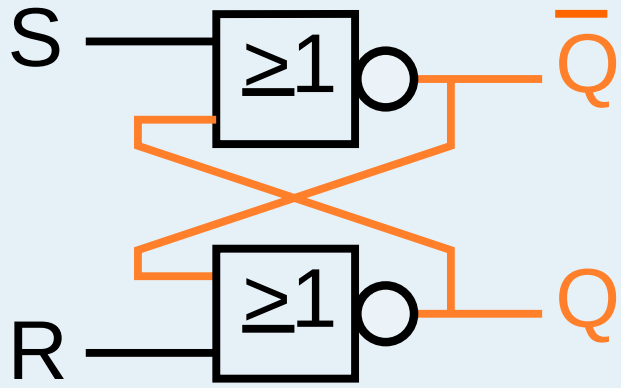


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

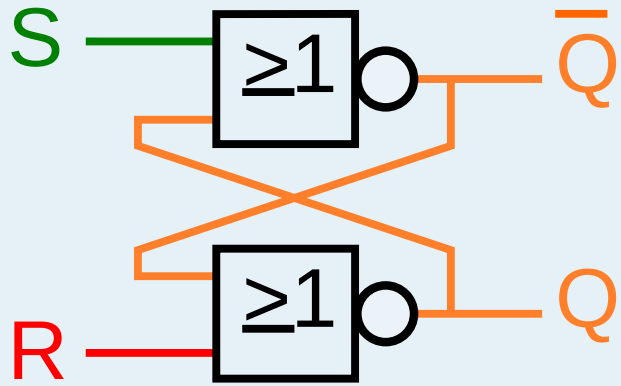


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

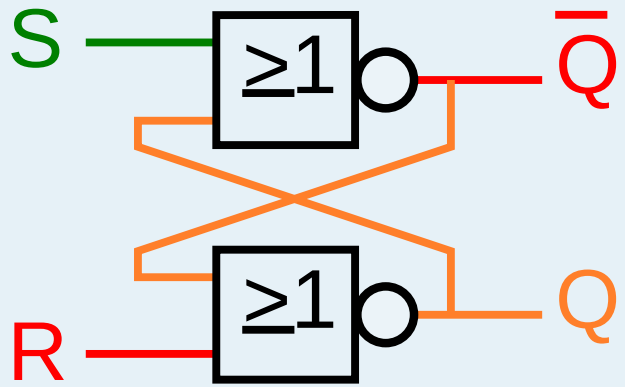


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

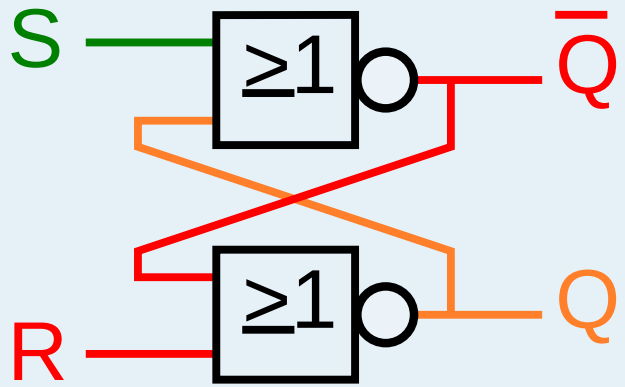


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

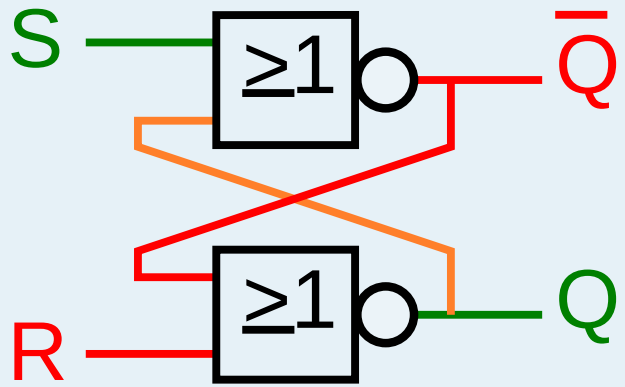


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

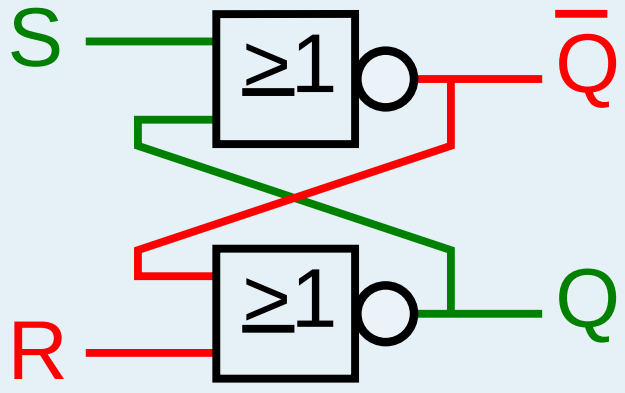


FIGURE – Un bit peut être stocké via une bascule RS

Bascule

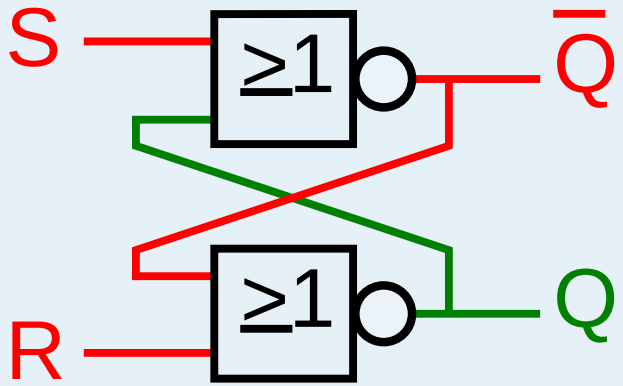


FIGURE – Un bit peut être stocké via une bascule RS

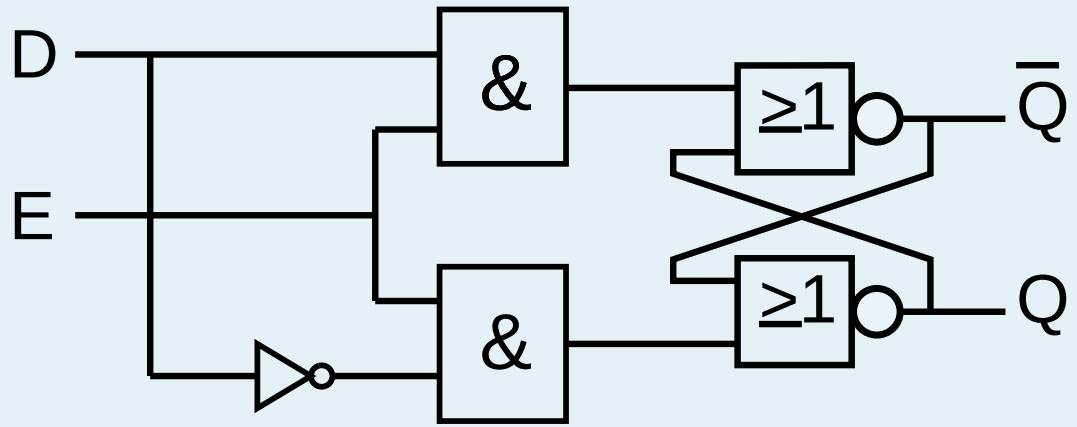


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

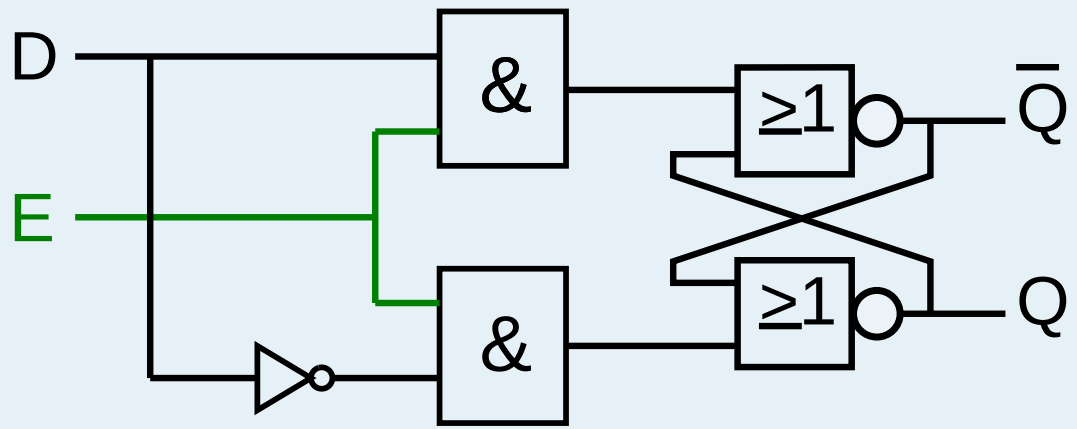


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

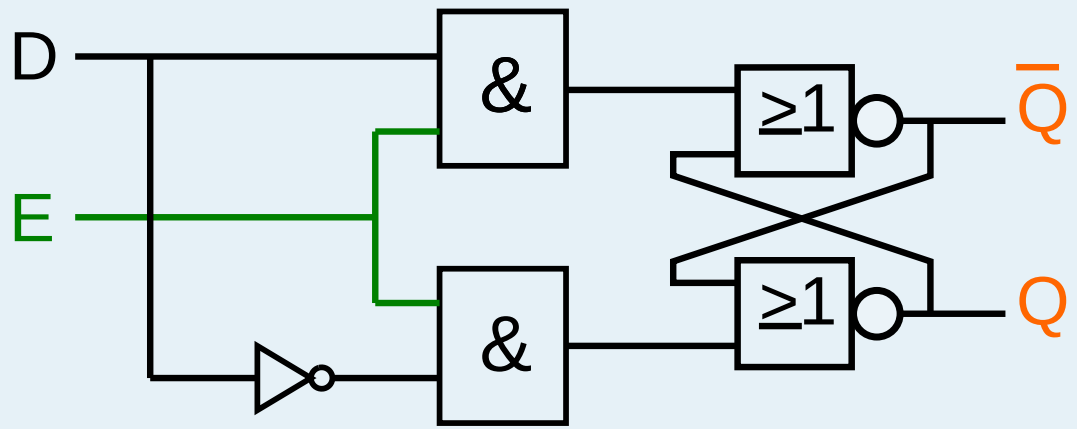


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

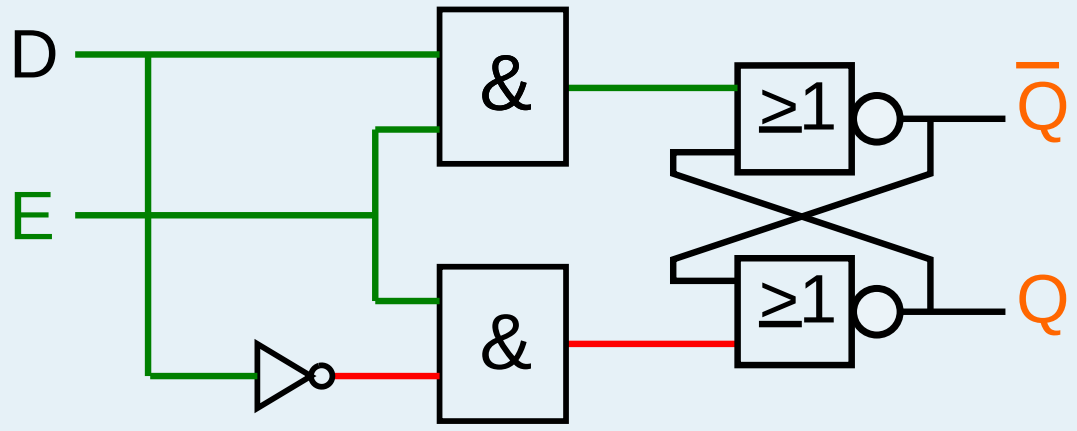


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

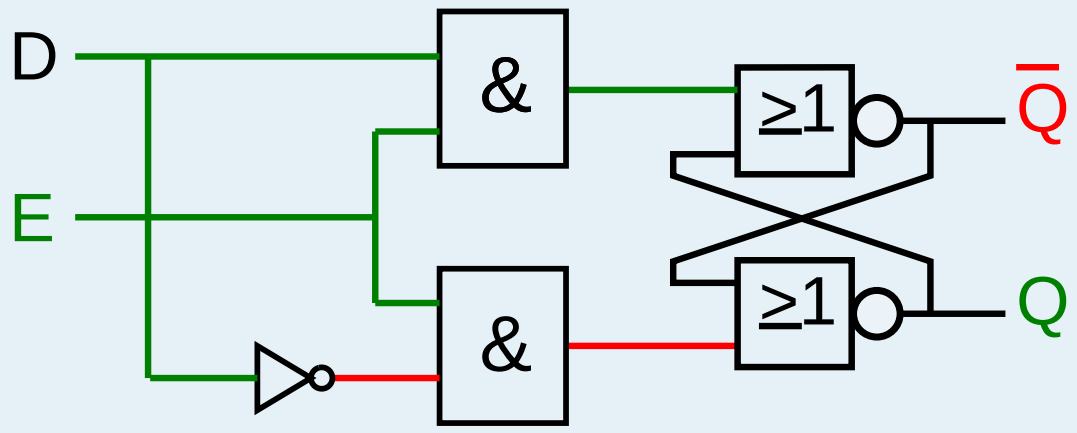


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

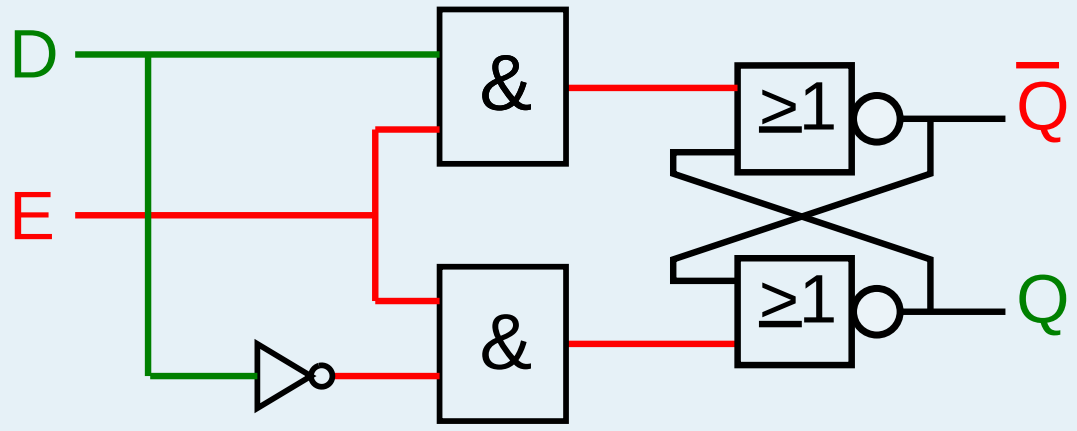


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

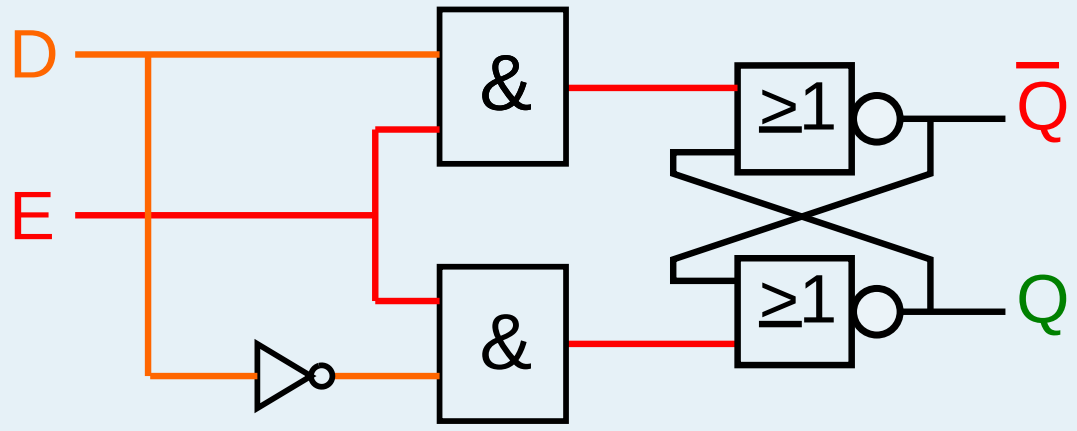


FIGURE – Bascule D

Comportement

- Si l'entrée E (Enable) est active, alors la sortie Q prend la valeur de l'entrée D.
- Sinon, la sortie Q reste inchangée.

Multiplexeur 2 vers 1

L'entrée e_0 ou e_1 est propagée sur la sortie q suivant la valeur de s_0.

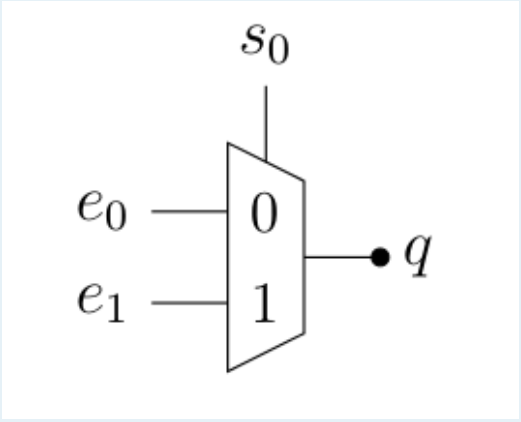
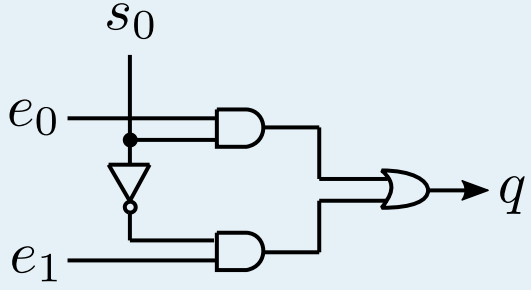


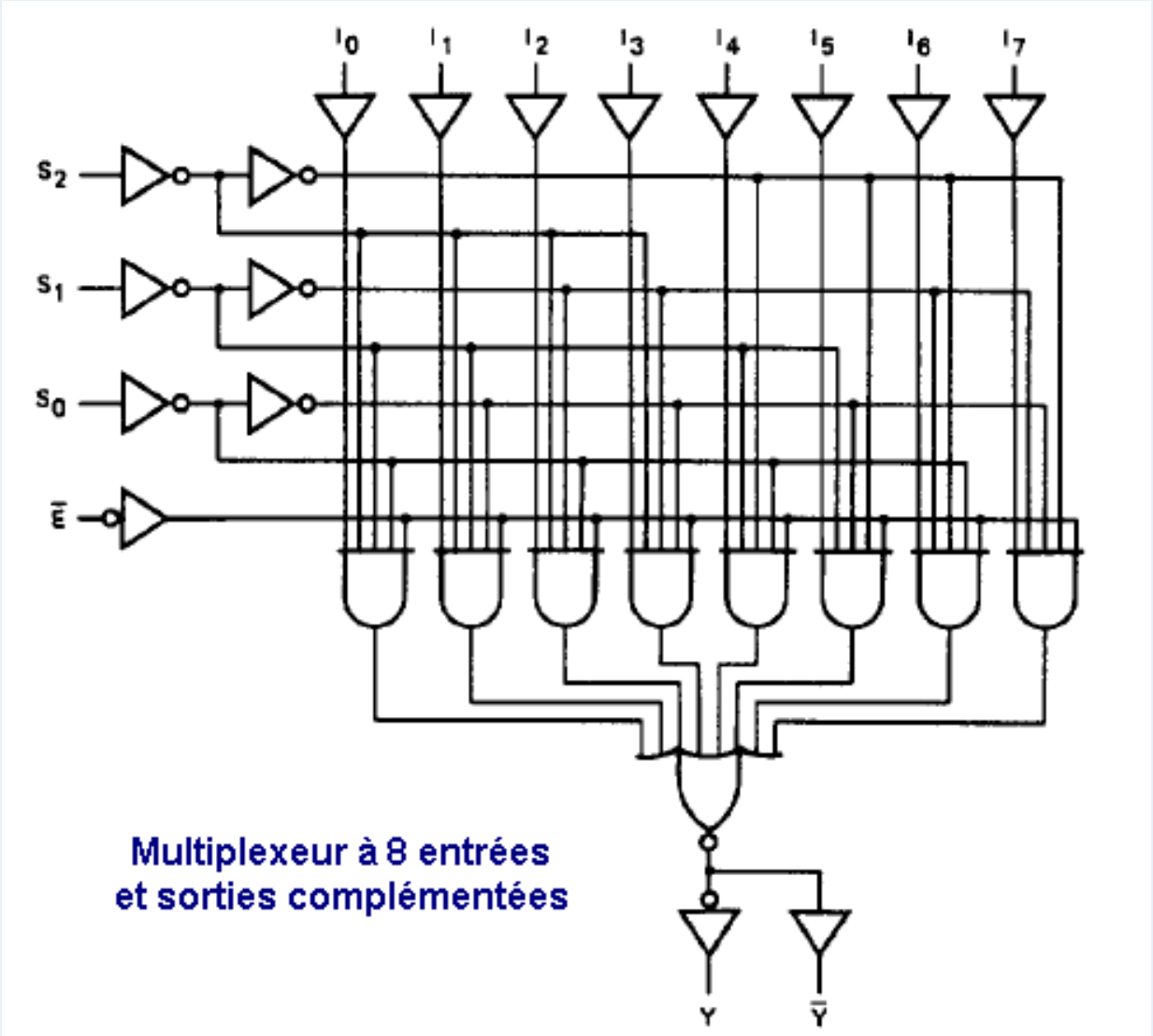
Table de vérité

s_0	q
0	e_0
1	e_1

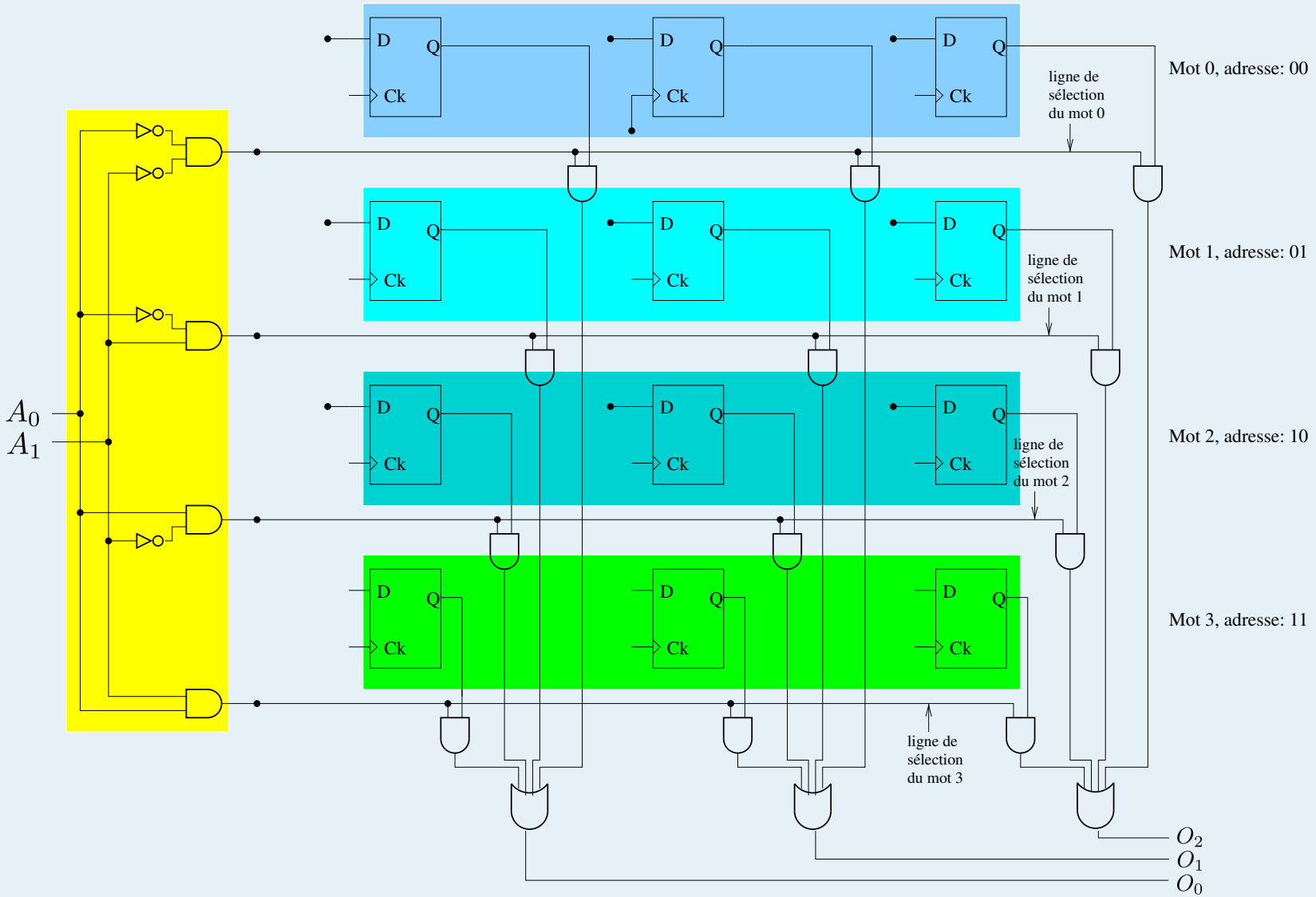
Avec des portes logiques



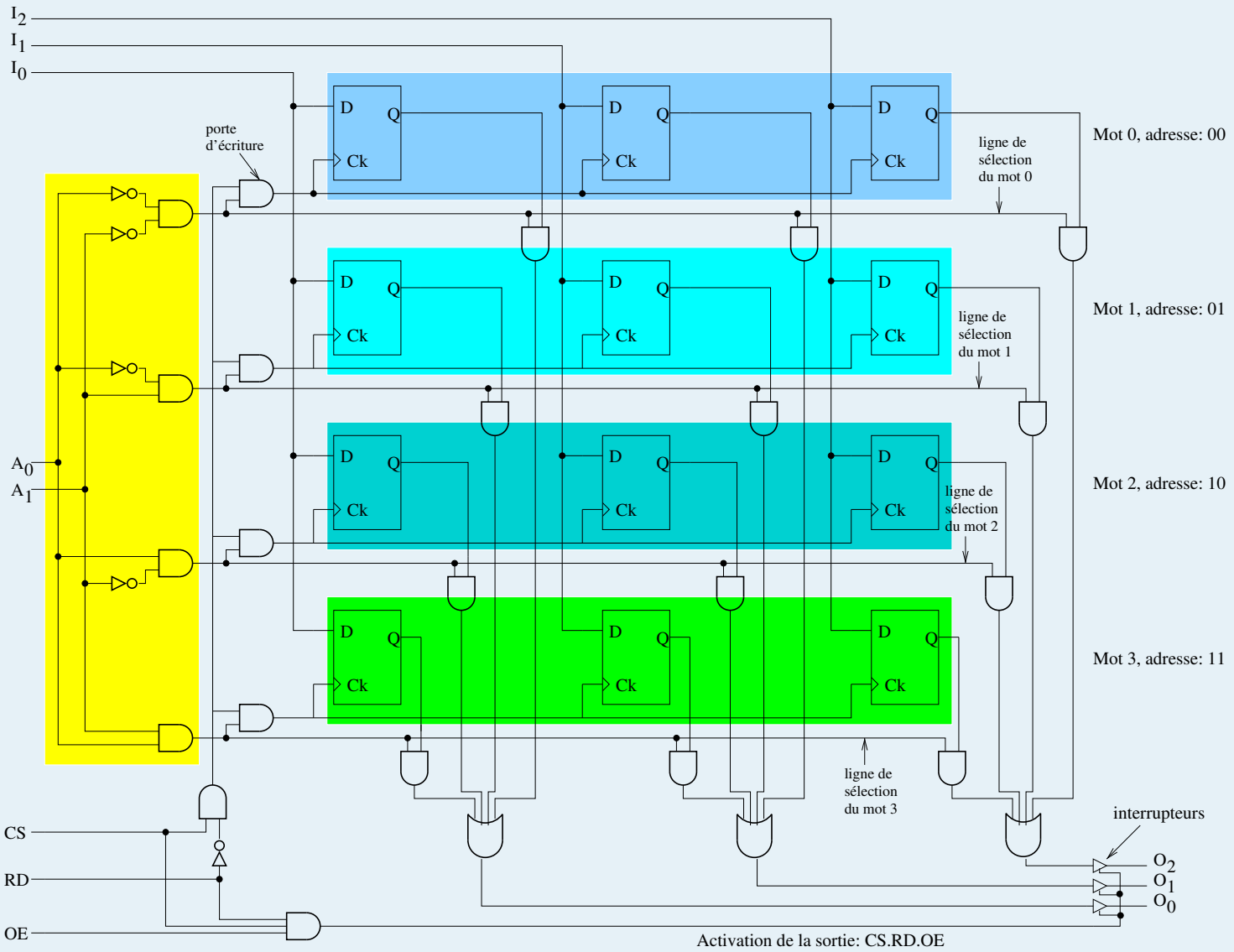
Multiplexeur 8 vers 1



Lecture mémoire de 4 mots de 3 bits



Lecture et écriture mémoire de 4 mots de 3 bits



Définition

Finalement, la mémoire est circuit capable de stocker des mots binaires.

Remarque

La mémoire peut être imaginée comme un tableau contenant des informations. Chaque case mémoire est repérée par un numéro

Définition

Le numéro d'une case est son adresse.

Exemple

Considérons une mémoire de 8 mots de 16 bits :

Adresses	Données
0x0	0x14D5
0x1	0x10D5
0x2	0x0000
0x3	0x1245
0x4	0x2544
0x5	0x1541
0x6	0xADF0
0x7	0xF457

A l'adresse 3, on a stocké la valeur 1245.

Remarque

Attention de ne pas confondre une adresse et le contenu de la case mémoire correspondante ...

Remarque

Une donnée peut être stockée dans plusieurs cases successives.

Exemple

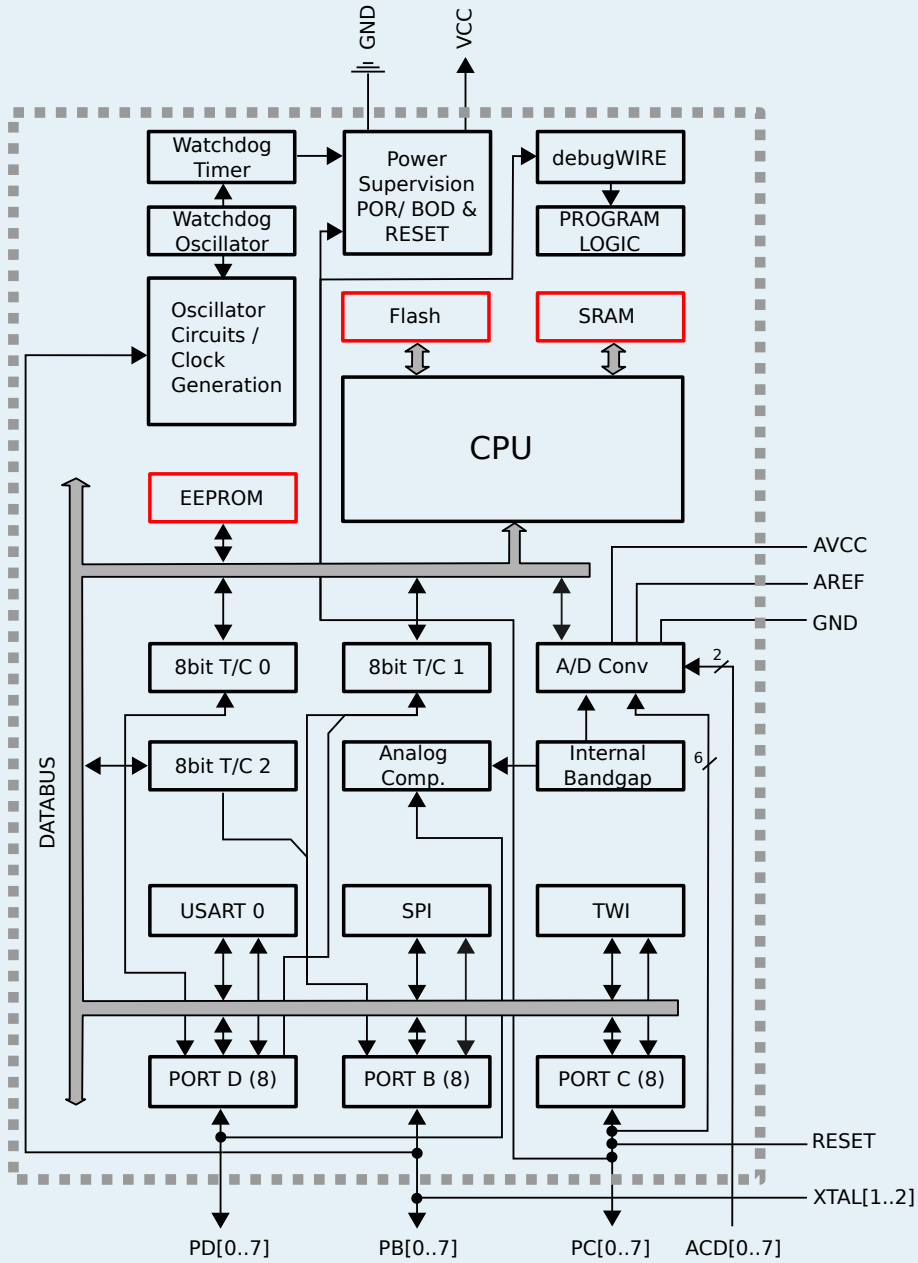
En effet, si on souhaite stocker une valeur sur 32 bits alors que chaque case ne peut accueillir que 8 bits, on “découpe” la valeur en 4 octets qu’on stocke dans 4 cases successives.

La mémoire suivante ne peut contenir que 2 mots de 32 bits :

Adresses	Données
0x0	0xD5
0x1	0xD5
0x2	0x00
0x3	0x45
0x4	0x44
0x5	0x41
0x6	0xF0
0x7	0x57

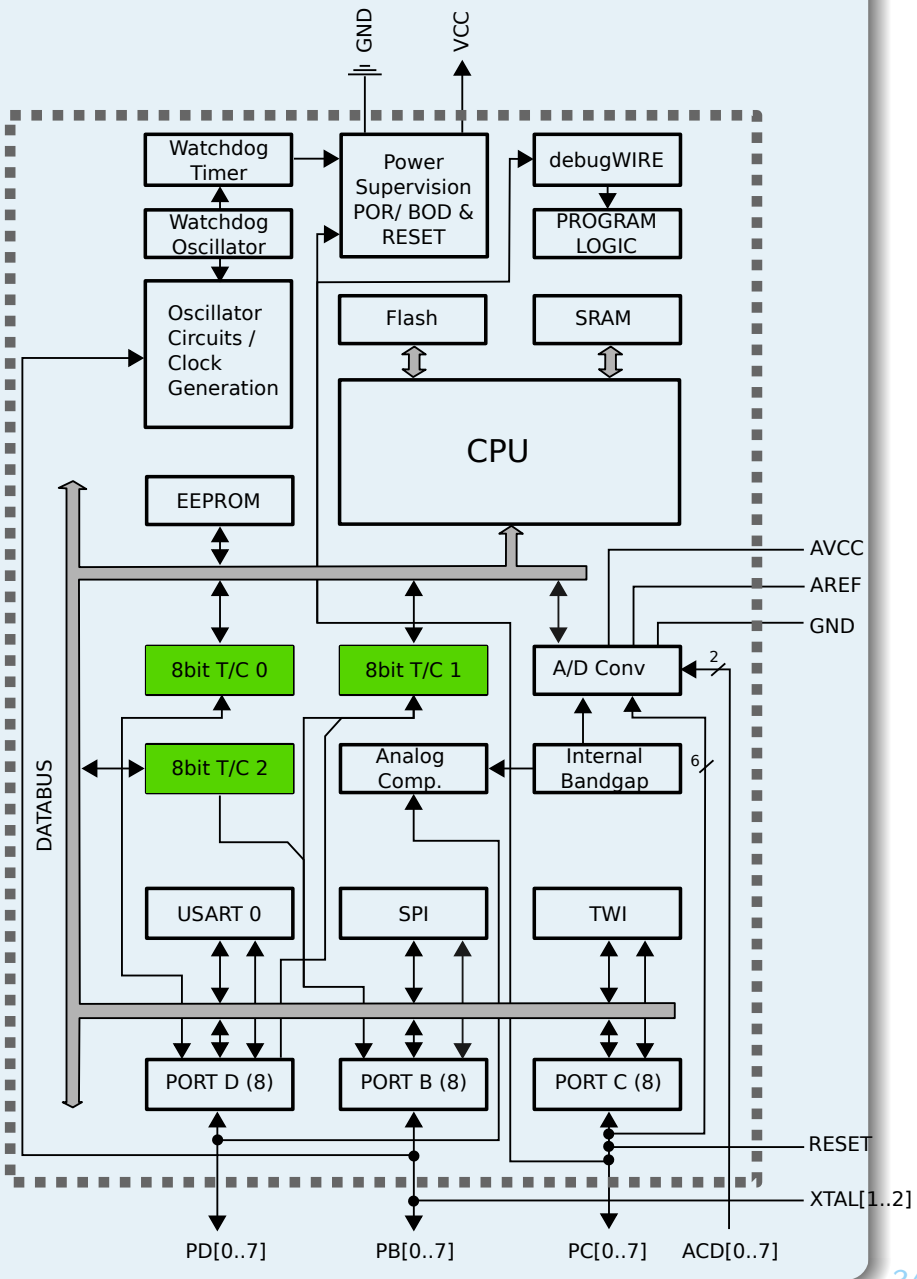
Mémoires

- Flash (32Ko) : mémoire contenant le programme
  - en lecture seule
  - non volatile
- SRAM (2Ko) :
  - données mémoires (volatile)
  - Valeurs temporaires, piles, ...
  - Espace limité !
- EEPROM (1Ko) : données à long terme, données sur le bus I/O



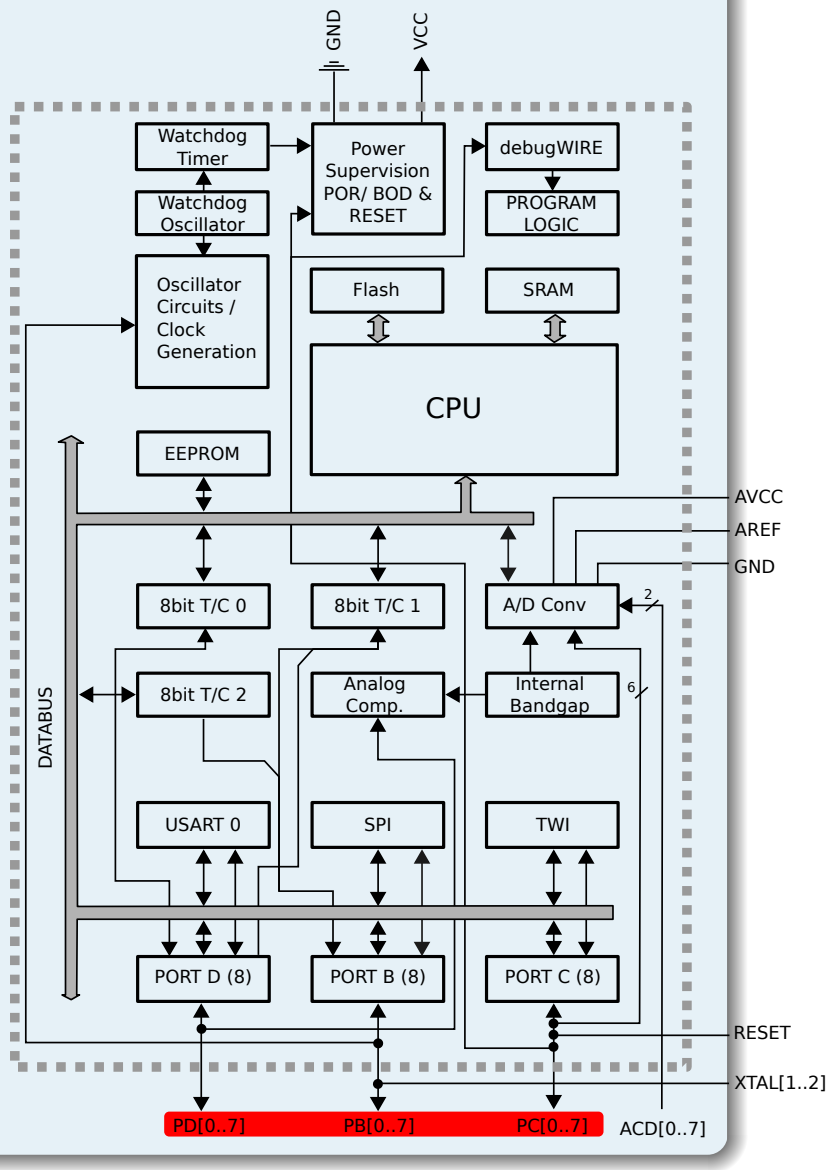
Trois timers

- Très flexible :
  - choix de fréquence d'horloge,
  - peuvent générer des interruptions
  - peuvent générer les signaux PWM
- Plus de détails dans un cours suivant . . .



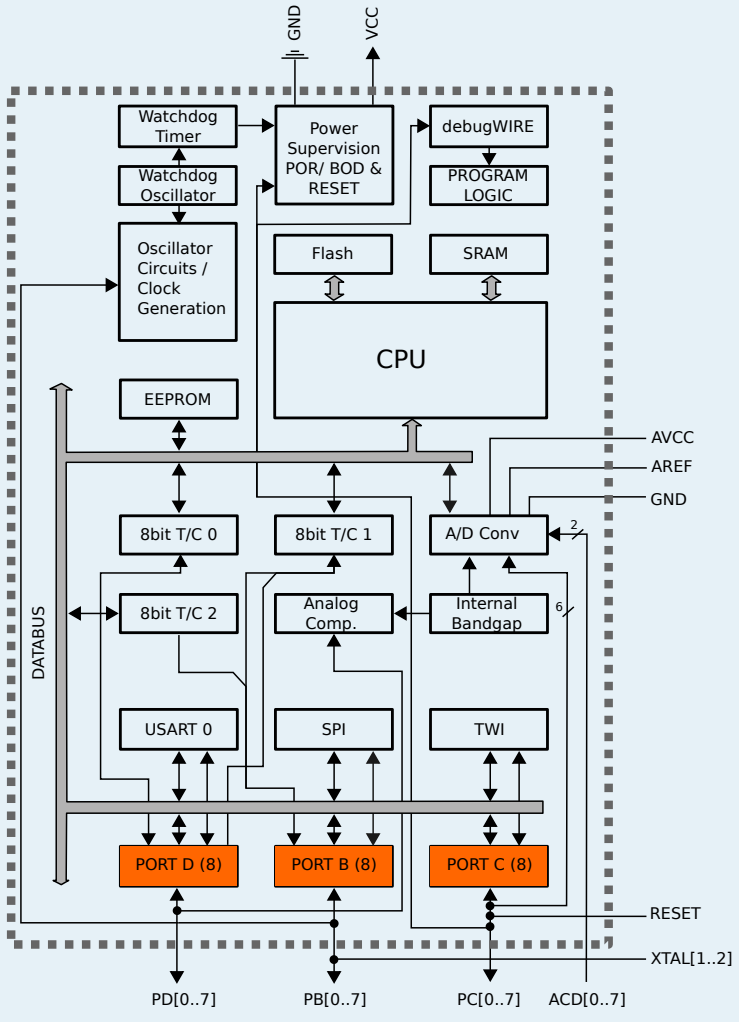
Broches d'entrées-Sorties

- Interaction avec le monde extérieur.
- On peut y brancher indirectement
  - des actionneurs (led, relais, ...),
  - ou bien des capteurs.



Interfaces vers les broches

- Chaque broche est directement paramétrable
  - choix de la direction,
  - choix de la valeur,
  - choix d'un pull-up interne ou non
- Certaines broches sont spéciales
  - analogique vs numérique,
  - horloges,
  - reset



Interfaces vers les ports I/O

- Trois ports 8-bit (B, C, D)
- Chaque port est controlé par 3 registres sur 8 bits.
  - Chaque bit de chacun de ces 3 registres correspond à une broche.
  - `DDRx` - registre de direction
    - permet de paramétrer une broche en tant qu'entrée (0) ou bien en tant que sortie (1).
  - `PINx` - valeur de l'entrée d'une broche
    - Lire un bit ce registre permet de connaitre la valeur de la broche correspondante.
  - `PORTx` - valeur de la sortie
    - Ecrire un bit dans ce registre permet de choisir la valeur d'une sortie.

Exemple d'écriture sur les 8 broches du port B

```
// Dans le setup(), paramétrage des 8 broches du portB en sortie
void setup() {
 DDRB=0xFF; // équivalent à DDRB=0b11111111,
            // toutes les broches du PORTB
            // sont des sorties
}

// Ecriture de 8 valeurs binaires sur le PortB
void loop() {
 ...
 PORTB=0x9F; // équivalent à PORTB=0b10011111,
             // les broches 6 et 5 sont à un niveau bas
             // les autres broches à un niveau haut
 ...
}
```

Exemple de lecture de la broche 3 du port B

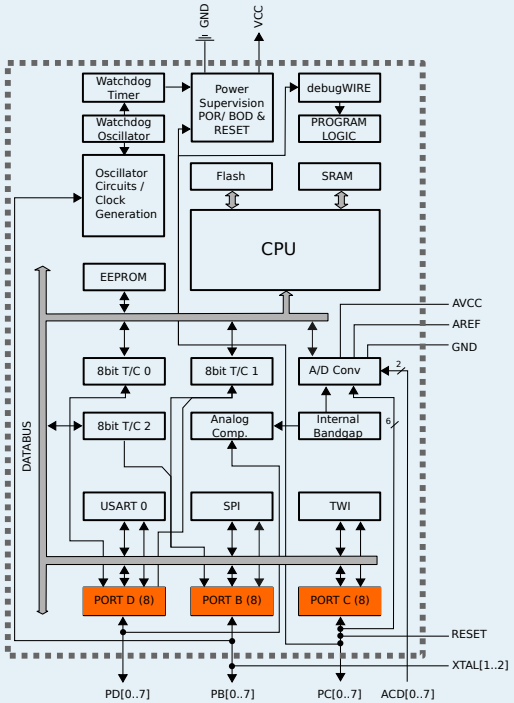
```
// Dans le setup(), paramétrage des 8 broches du portB
void setup() {
 DDRB=0xF0; // équivalent à DDRB=0b11110000,
            // les 4 premières broches du PORTB sont des sorties
            // les 4 autres des entrées
}

// Lecture du 3 ieme bit du PortB
void loop() {
 ...
 if((PINB&0x04)==0)  // 0x04 est équivalent à 0b00000100,
 {
   // L'entrée correspondante au 3ieme bit du portB
   // est à un niveau bas.
   // Ici se trouve le code à exécuter dans ce cas.
 }
}
```

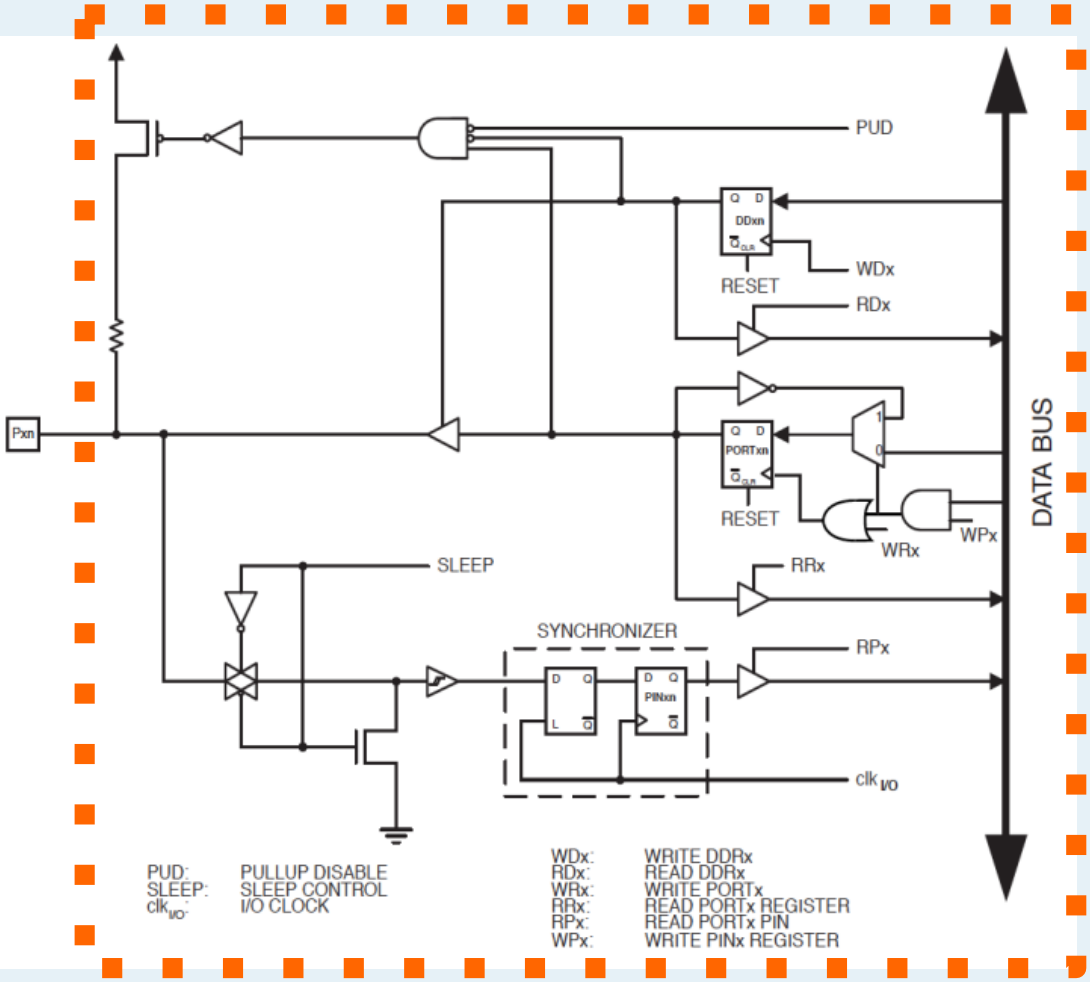
Remarque

L'instruction `(PINB&0x04)==0)` sera expliquée dans la section “masquage”.

Explications

Comment DDRx permet-il de configurer une broche en entrée ou sortie ?





Explications

Comment le registre DDRx permet-il de configurer une broche en entrée ou sortie ?

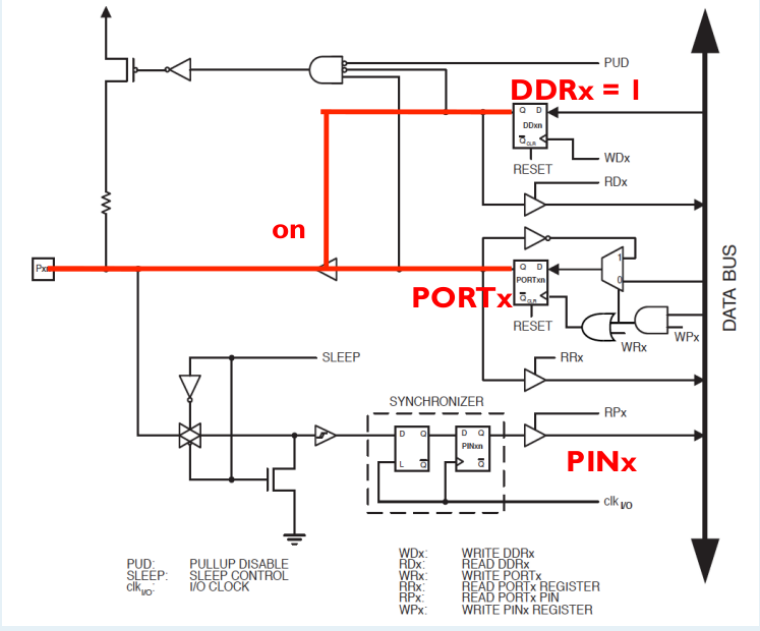


FIGURE – La broche est ici configurée en mode sortie.

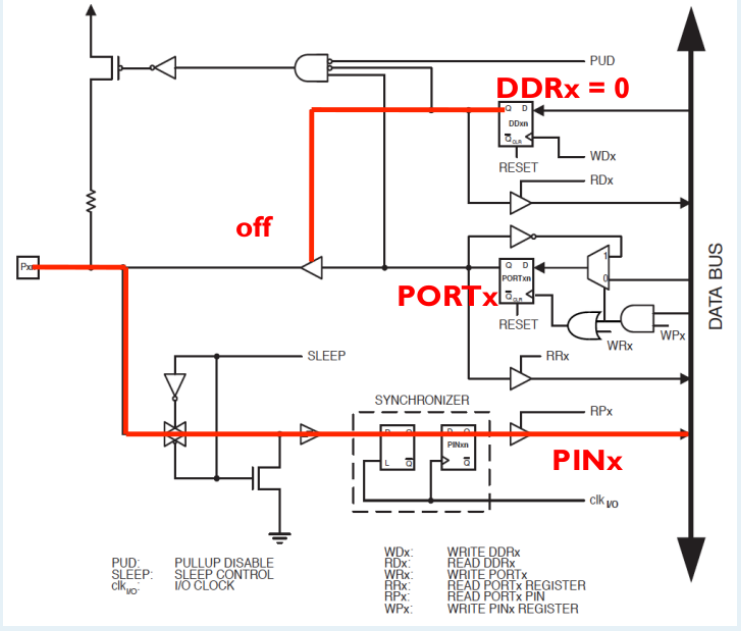


FIGURE – La broche est ici configurée en mode entrée.

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$								

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0							

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0						

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5					

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5	a_4				

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5	a_4	a_3			

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5	a_4	a_3	0		

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5	a_4	a_3	0	a_1	

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5	a_4	a_3	0	a_1	0

Masquage logique

Le masquage est une opération de logique combinatoire dans un groupe de bits un sous-ensemble de bits à conserver, les autres étant forcés à une valeur choisie.

Exemple masquage “ET logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	1	1	0	1	0
résultat $a \& m$	0	0	a_5	a_4	a_3	0	a_1	0

Proposition

Un masquage “ET logique” permet d’écraser certains bits à 0 et de conserver les autres.

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a \mid m$								

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a \mid m$	a_7							

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a\|m$	a_7	a_6						

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a\|m$	a_7	a_6	1					

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a\|m$	a_7	a_6	1	a_4				

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a\|m$	a_7	a_6	1	a_4	1			

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a\|m$	a_7	a_6	1	a_4	1	1		

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a\|m$	a_7	a_6	1	a_4	1	1	1	

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a \mid m$	a_7	a_6	1	a_4	1	1	1	a_0

Exemple masquage “OU logique”

octet a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
masque m	0	0	1	0	1	1	1	0
résultat $a \mid m$	a_7	a_6	1	a_4	1	1	1	a_0

Proposition

Un masquage “OU logique” permet d’écraser certains bits à 1 et de conserver les autres.

Exemple en c

```
char a0 = 0xF3;
     a0 |= 0b08;
```

La seconde instruction permet de spécifier que le 4ième bit de la variable a0 à 1 tout en laissant les autres inchangés.

Exemple en c

```
char a0 = 0xF3;
    a0 |= 0b08;
```

La seconde instruction permet de spécifier que le 4ième bit de la variable a0 à 1 tout en laissant les autres inchangés.

Explication en binaire

```
  char a0 = 0b11110011;
       a0 |= 0b00001000;
// a0 vaut     11111011
```

Instruction de lecture du port B

Le booléen `(PINB & 0x04)==0` est vrai si et seulement si la troisième broche du port B est à un niveau bas.

Explication en binaire

PINB	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0x04	0	0	0	0	0	1	0	0
PINB&0x04	0	0	0	0	0	a_2	0	0