

Enseignes et afficheurs à LED

Les entrées-sorties

Pierre-Yves Rochat

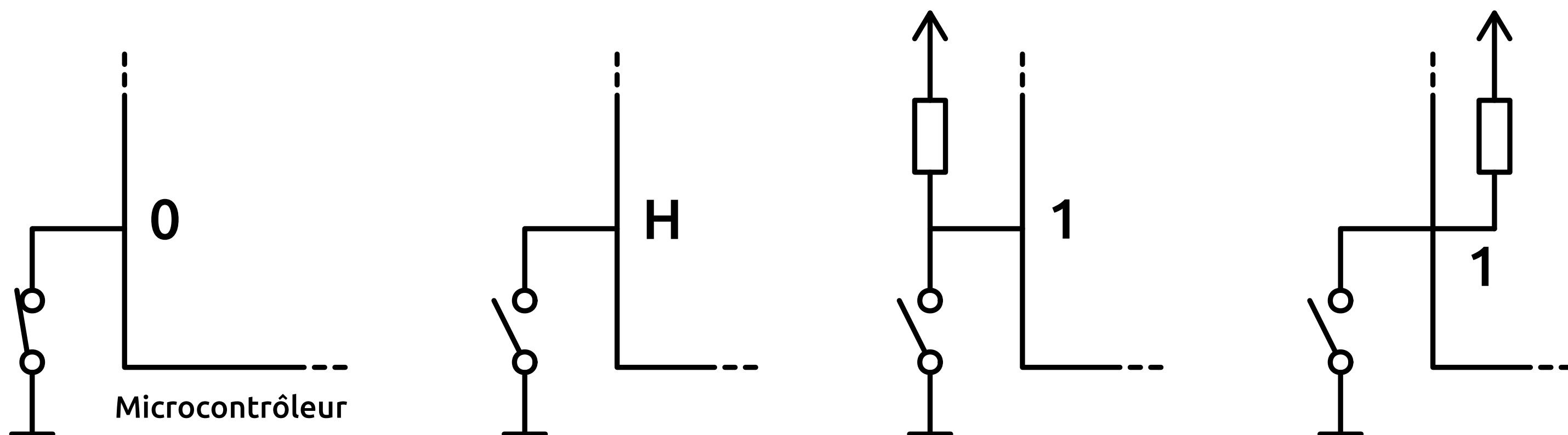
- Rôle des broches en entrée et en sorties
- Ports et registres sur AVR et MSP430
- Opérations logiques sur champs de bits
- Écritures des constantes

Broches en entrée et en sorties

- GPIO : *General Purpose Input Output*
 - Arduino : choix par `pinMode()`
 - Lecture par `digitalRead()`
 - Écriture par `digitalWrite()`
 - Simple... mais pas toujours optimal
-
- Accès à une seule broche à la fois
 - Temps d'exécution important
 - Taille mémoire peu optimale

Rôle des broches

- Entrée à haute impédance
- Entrée avec une résistance de tirage vers le haut : *pull-up*
- Entrée avec une résistance de tirage vers le bas : *pull-down*
- Sortie à 0
- Sortie à 1



Les ports et leurs registres

- Les broches sont regroupées par **ports**
- Les ports ont souvent 8 bits, parfois 16 ou 32 bits
- Un port peut être incomplet sur un modèle donné de microcontrôleur
- Les noms des ports dépendent des familles de microcontrôleurs
- On accède aux broches et à leur rôle par des **registres**
- **PIC** : PORTA — TRIS
- **AVR** : PORTA — DDRA — PINA
- **MSP430** : P1DIR — P1OUT — P1IN — P1REN

Les registres sur les AVR

- Sur les AVR, les ports s'appellent PORTA, PORTB, ...
- Les 8 broches du PORTA s'appellent PA0, PA1... PA7

3 registres sont utilisés pour piloter chaque port :

- **DDRA** *Data Direction Register*
- **PORTA** : registre de sortie
- **PINA** : donne l'état de chaque broche

DDR	PORT	Rôle de la broche
0	0	Entrée
0	1	Entrée avec pull-up
1	0	Sortie à 0
1	1	Sortie à 1

Bit 6



Les registres sur les MSP430

- Sur les AVR, les ports s'appellent P1, P2, ...
- Les 8 broches de P1 s'appellent P1.0, P1.1... P1.7

4 registres sont utilisés pour piloter chaque port :

- **P1DIR** *Data Direction Register*
- **P1OUT** : registre de sortie
- **P1IN** : donne l'état de chaque broche
- **P1REN** : enclenche une résistance de tirage

REN	DIR	OUT	Rôle de la patte
0	0	0	Entrée
1	0	1	Entrée avec Pull-up
1	0	0	Entrée avec Pull-down
0	1	0	Sortie à 0
0	1	1	Sortie à 1

Lecture et écriture sur un port

- Initialisations : `P1DIR = 0b01000001;`
- Lecture : `variable = P1IN;`
- Écriture : `P1OUT = valeur;`
- Comment agir sur un bit à la fois ?
- Grâce aux opérateurs logique du C !
- Le **OU** logique : `|`
- Le **ET** logique : `&`
- L'**inversion** logique : `~`

Mise à un de bits

- $P10UT \mid = 0b01000000;$

Avant :

x7	x6	x5	x4	x3	x2	x1	x0
----	----	----	----	----	----	----	----

0	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

OU

Après :

x7	1	x5	x4	x3	x2	x1	x0
----	---	----	----	----	----	----	----

- Également sur plusieurs bits : $P10UT \mid = 0b01000001;$

Mise à zéro de bits

- `P10UT &= 0b10111111;`

Avant :

x7	x6	x5	x4	x3	x2	x1	x0
----	----	----	----	----	----	----	----

1	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---

ET

Après :

x7	0	x5	x4	x3	x2	x1	x0
----	---	----	----	----	----	----	----

- Également sur plusieurs bits : `P10UT &= 0b10111101;`

Écriture plus lisible des constantes

- `P10UT |= 64;`
- `P10UT |= 0x40;`
- `P10UT |= 0b01000000;`
- `P10UT |= (1<<6);`
- Avec l'opérateur d'inversion : `P10UT &=~(1<<6);`
- *bit set* : `P10UT |= (1<<6);`
- *bit clear* : `P10UT &=~(1<<6);`

Inversion d'un bit par OU exclusif

- Le C offre un opérateur pour le OU exclusif : ^
- `P10UT ^= (1<<6);` : inverse le bit 6

Utilisations de `#define`

```
1 #define ClockSet P10UT |= (1<<5)
2 #define ClockClear P10UT &=~(1<<5)
3
4 #define LedRougeOn P10UT |= (1<<0)
5 #define LedRougeOff P10UT &=~(1<<0)
6 #define LedRougeToggle P10UT ^= (1<<0)
```

Les entrées-sorties

- Rôle des broches en entrée et en sortie
- Ports et registres sur AVR et MSP430
- Opérations logiques sur champs de bits
- Écritures des constantes