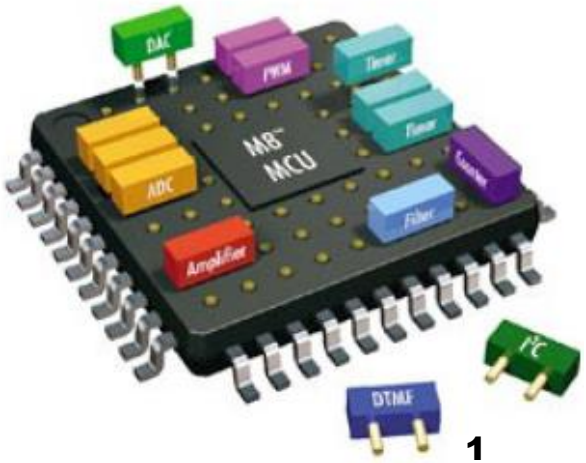
Microcontrôleur AVR

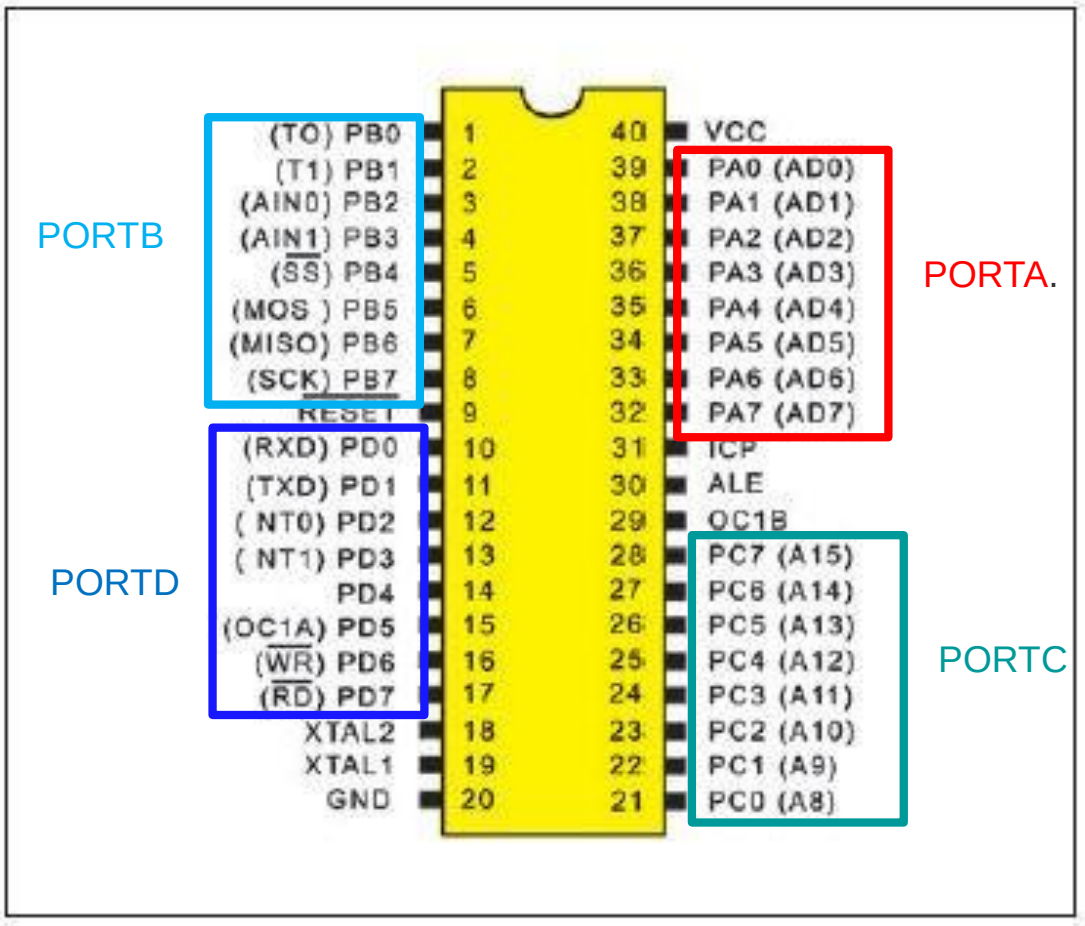
Informatique embarquée

1ère année master RSD

Dr: k.Barka



1. Introduction (microcontrôleur AVR)



Brochage du AT90S8515 (AVR).

1. Les Ports

Le nombre de PORTs disponibles sur les AVR dépend de la série. À part la série TINY qui sont des composants de petites tailles, les AVRs de la série Mega comportent au moins les PORTs A, B, C et D.

- **Port A (PA7…PA0)** : C'est un port de I/O bidirectionnel.
- **Port B (PB7…PB0)** : C'est un port de I/O bidirectionnel.
- **Port C (PC7…PC0)** : C'est un port de I/O bidirectionnel.
- **Port D (PD7…PD0)** : C'est un port de I/O bidirectionnel.

1. Les registres

Chaque port d'I/O est piloté à travers l'utilisation de trois registres qui sont appelés

- Data Register : **PORTx**, Le registre des données
- Data Direction Register: **DDRx** qui indique la direction de la donnée, In (entrée) ou Out (sortie)
- Input Pins Address: **IPINx**

Exemple X=A

"PORTA" (adresse hexadécimale H3B),
"DDRA"(adresse H3A),
"PINA" (adresse H39).

	PORT	DDR	PIN
PORTA	$H3B	$H3A	$H39
PORTB	$H38	$H37	$H36
PORTC	$H35	$H34	$H33
PORTD	$H32	$H31	$H30

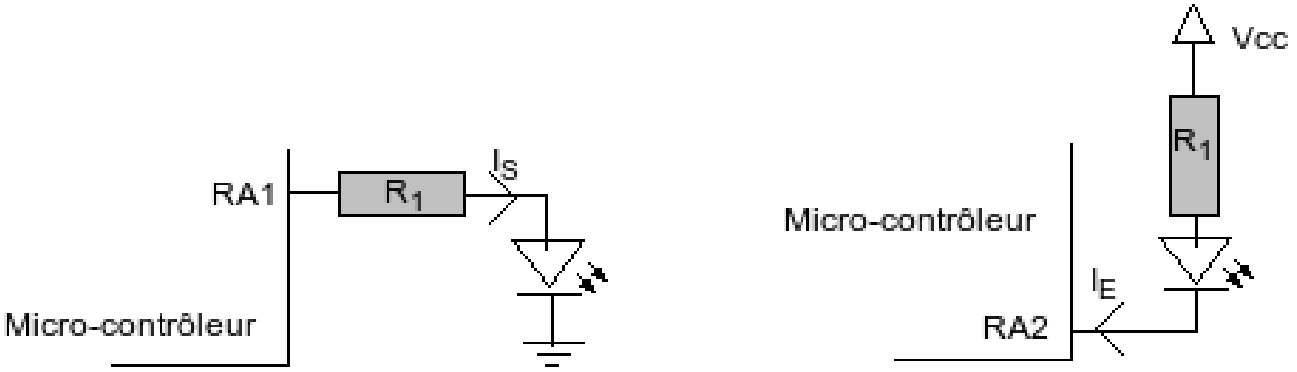
Adresses hexadécimales des ports d'I/O.

1. Les connexions électriques à partir des PORTs

Le but de cette section est d'étudier quelques connexions électriques à partir des PORTs d'un microcontrôleur. Nous nous contenterons des **LED** et des **boutons poussoirs**.

1. Connecter et dimensionner des LEDs

Il y a deux façons typiques pour connecter des LEDs :

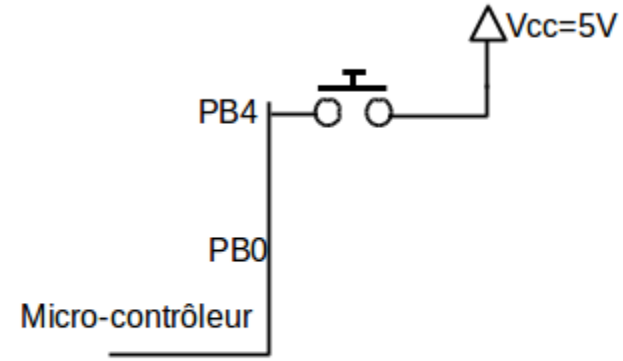


À gauche c'est le microcontrôleur qui est à l'origine du courant. À droite c'est lui qui reçoit le courant passant par la LED et provenant d'une alimentation. Évidemment, il faut un **1** logique sur le bit **PA1** pour allumer la LED de gauche et un **zéro** logique sur le bit PA2 pour allumer la LED de droite

1. Les connexions électriques à partir des PORTs

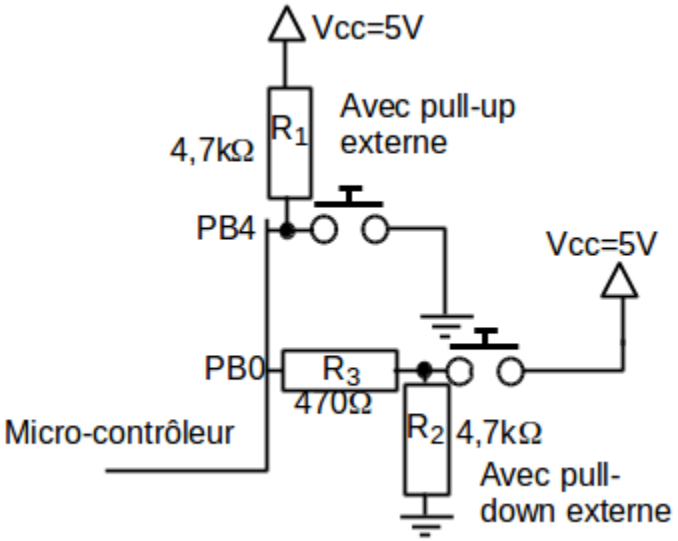
1. Connecter des boutons poussoirs

PB4 = 1 logique (bouton lâché ou appuyé)



Version intuitive mais ne fonctionne pas toujours !

PB4= 1 bouton lâché sinon 0
PB6= 1 bouton appuyé sinon 0



1. Les connexions électriques à partir des PORTs

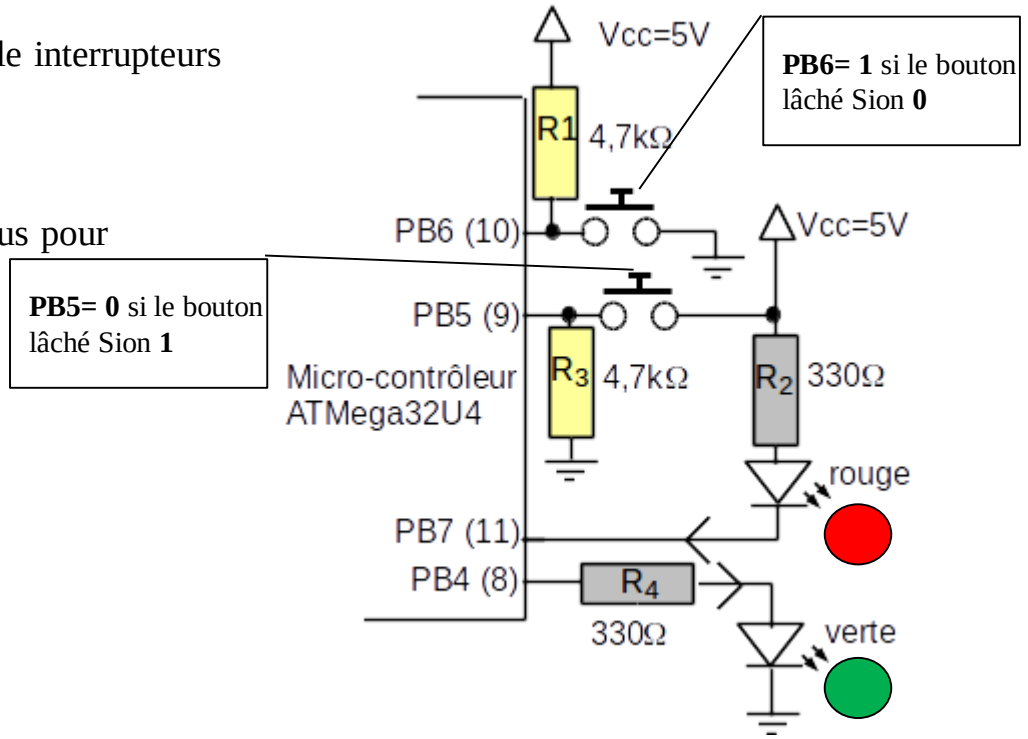
1. Exemple 1

On désire écrire un programme C qui ne fait rien si on n'appuie sur aucun bouton poussoir, fait clignoter la **LED rouge** si l'on appuie sur un bouton, fait clignoter la **LED verte** si on appuie sur l'autre bouton, et les deux LEDs si l'on appuie sur les deux boutons.

a) Donner les 4 valeurs possibles de la variable interrupteurs avec l'instruction

interrupteurs = PINB & 0x60;

b) Compléter alors le morceau de code ci-dessous pour en faire un programme :



1. Solution

a) Nous appelons les interrupteurs par le nom des broches auxquels ils sont connectés.

- PB6 et PB5 lâchés : 0x40
- PB6 lâché et PB5 appuyé : 0x60
- PB5 lâché et PB6 appuyé : 0x00
- PB6 et PB5 appuyés : Ox20

1. Les connexions électriques à partir des PORTs

1. Solution

b)

```
#undef F_CPU
#define F_CPU 25000000UL
#include "util/delay.h"
int main() {
  unsigned char leds = 0x00;// variable de gestion des LEDs
// setup()
  DDRB |= 0x90; //PB7 et PB4 en sortie
// loop()
  while(1) {
    // boucle d'attente ATTENTION PB6=1 et PB5=0 au repos
    while(interrupteurs == 0x20)
       interrupteurs = PINB & 0x60;
    switch(interrupteurs) {
       case 0x60 : PORTB = leds^0x10;break;
       case 0x00: PORTB = leds^0x80;break;
       case 0x20 : PORTB = leds^0x90;break; // les deux
    }
    _delay_ms(1000);
  } // while(1)
  return 0;
} // main
```

Les connexions électriques à partir des PORTs

1. Exemple 2

Une partie matérielle est constituée de deux afficheurs sept segments multiplexés. Les sept segments sont commandés par le **PORTC**, tandis que les commandes d'affichages sont réalisée par les bits b0 et b1 du **PORTB**. Un schéma de principe est donné ci-après.

1- calculer les valeurs dans un tableau `"unsigned char SEGMENT[] = {0x3F,...};` pour un affichage des chiffres de 0 à 9.

2- réaliser une fonction responsable du transcodage :

```
unsigned char Display(unsigned char no) {
unsigned char Pattern;
unsigned char SEGMENT[] = {0x3F,....
```

3- Réaliser le programme **main()** responsable de l'initialisation de l'interruption qui doit avoir lieu toutes les 10ms qui compte de 00 à 99 toutes les secondes environ (avec un "_delay_ms(1000);")

4- Réaliser enfin l'interruption qui affichera tantôt les dizaines, tantôt les unités.