Chapitre 28 – La face cachée des fonctions pinMode, digitalWrite et digitalRead

La face cachée de la fonction pinMode

Les entrées-sorties numériques sont soit des entrées, soit des sorties et il est nécessaire de les déclarer en entrée ou en sortie dans la fonction setup. Ceci est extrêmement simple avec Arduino puisqu'il suffit d'utiliser la fonction pinMode qui va s'occuper de tout. Néanmoins, ce cours concerne les microcontrôleurs en général et pas seulement Arduino qui n'est utilisé qu'en tant qu'outil de programmation. Nous allons donc voir dans ce chapitre ce qui se passe réellement dans le microcontrôleur quand on utilise la fonction pinMode.

Organisation des broches du microcontrôleur

Les microcontrôleurs ont la particularité d'avoir des broches multiplexées, ce qui signifie **qu'une même broche peut avoir plusieurs utilisations possibles**. Ceci est un moyen de ne pas multiplier le nombre de broches, donc d'avoir des composants plus petits. Les E/S du module Uno sont bien évidemment reliées aux broches d'E/S du microcontrôleur. La figure 28.1 permet de voir la correspondance entre les E/S du module Uno et les broches du microcontrôleur ATmega328P.

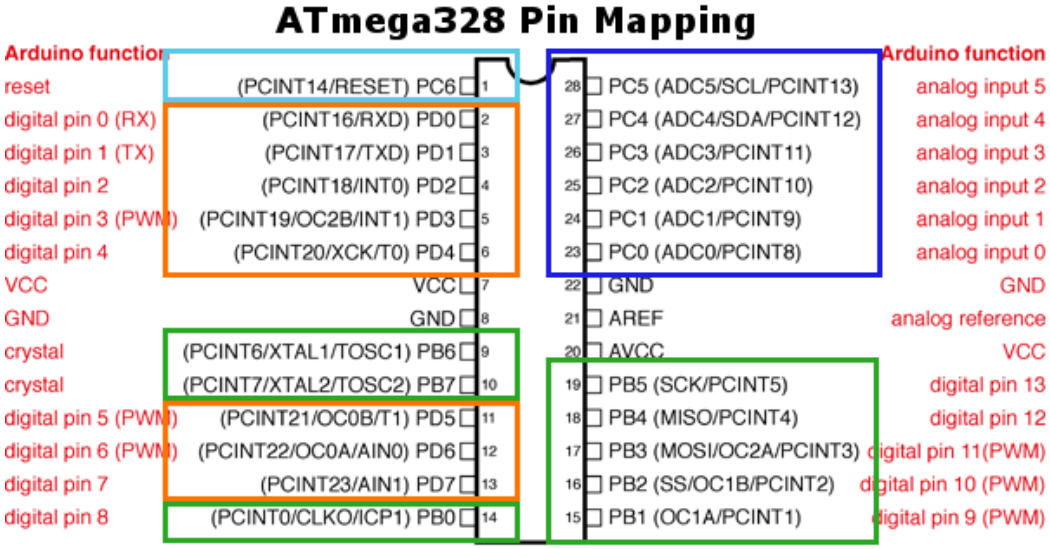

Figure 28. 1

La première chose qu'on peut remarquer, c'est que les broches d'E/S du microcontrôleur sont organisées en **port** d'entrées-sorties. Un port est un ensemble de ligne d'E/S. Par exemple, pour l'ATmega328P, nous trouvons trois ports appelés respectivement B, C, et D. Les ports B et D ont huit lignes numérotées 0 à 7. Le port C est un peu particulier : c'est un port à 6 lignes (0 à 5) mais il existe aussi une septième ligne PC6 qui sert au reset du microcontrôleur. Pour ne pas compliquer trop les choses à ce niveau, nous retiendrons que le port C a 6 lignes (PC5:0) et que les ports B et D ont 8 lignes (PB7:0 et PD7:0). Sur la figure 28.1, le port B est entouré en vert, le port C est entouré en bleu foncé, la broche reset (PC6) est entourée en bleu clair et le port D est entouré en orange.

Face aux broches du microcontrôleur, on trouve la correspondance en rouge des broches d'E/S du module Uno. On voit par exemple que la sortie 13 du module Uno correspond à la broche 19 du

microcontrôleur ou encore à la ligne 5 du port B. On voit aussi que les six entrées analogiques A0 à A5 sont reliées au port C.

Port et registre de contrôle

À chaque port correspond un **registre de direction** appelé DDR (DDRB, DDRC ou DDRD). Un registre est un espace mémoire du microcontrôleur où celui-ci peut lire ou écrire différentes valeurs. Comme la mémoire d'un microcontrôleur est organisée sous forme d'octets (ensemble de 8 bits, chaque bit ayant deux valeurs possibles qui sont 0 ou 1), les valeurs qu'on peut avoir dans un octet vont de 0 (tous les bits à 0) à 255 (tous les bits à 1). Vous pouvez utiliser la calculette de Windows pour vérifier que l'octet binaire 00000000 vaut 0 en décimal alors que l'octet (binaire) 11111111 vaut 255 (voir le chapitre 11).

Pour mettre une ligne d'E/S d'un port d'E/S en entrée, il suffit que le bit du registre de direction DDR correspondant au numéro de la ligne soit égal à 0. Pour que la ligne d'E/S d'un port soit configurée en sortie, il faut que le bit du registre de direction DDR correspondant au numéro de la ligne soit égal à 1.

Par exemple, la ligne d'E/S N° 13 du module Arduino correspond à la 5ème ligne du port d'E/S appelé PORT B. Pour que cette ligne soit en sortie, il faut donc que le bit N° 5 du registre de direction DDRB soit égal à 1. À la place de la ligne d'instruction pinMode (13, OUTPUT) ; on peut écrire DDRB = 0b00100000 ;

0b00100000 est la **valeur binaire** ; on voit tout de suite quels sont les bits à 0 et quels sont ceux à 1. Ici, tous les bits sont à 0 excepté le bit N° 5 qui est à 1 (on rappelle que les bits se numérotent de 0 à 7 en commençant par le plus à droite).

Exemple : ouvrez le programme Blink et remplacez dans la fonction setup la ligne qui contient pinMode par la ligne suivante :

DDRB = 0b00100000 ;

Téléversez le programme dans le microcontrôleur ; la LED clignotera de la même façon.

Nous serons amenés dans ce cours à parler des différents registres de contrôle du microcontrôleur et parfois à modifier leurs valeurs. Heureusement pour nous, le compilateur de l'IDE connait les noms des registres (PORTB et DDRB par exemple). Lorsque le compilateur tombe sur l'instruction :

pinMode (13, OUTPUT) ;

il sait qu'il doit positionner à 1 le bit N° 5 du registre de direction associé au PORT B, ou ce qui revient au même, écrire 00100000 dans le registre DDRB. Vous pouvez aussi le faire à sa place comme dans l'exemple traité en modifiant le programme Blink.

Cette méthode est plus compliquée, mais elle peut se révéler plus rapide quand on traite plusieurs lignes d'E/S. Néanmoins, cela a été l'occasion de parler des registres de contrôle et vous verrez qu'il y en a plein d'autres. Pour chaque registre, on peut manipuler des bits et les positionner à 0 ou à 1. On verra plusieurs méthodes pour cela.

À retenir sur la fonction pinMode :

- Les lignes d'E/S doivent être initialisées en entrée ou bien en sortie.
- La fonction qui fait cela est pinMode.
- Les lignes d'E/S du module Uno sont raccordées aux broches d'E/S du microcontrôleur.
- Les broches d'E/S du microcontrôleur sont organisées en PORT et il y a trois ports possibles.

- Chaque PORT est associé à un registre de direction appelé DDR.
- La valeur du bit du registre de direction détermine si la ligne correspondante est en entrée (bit à 0) ou en sortie (bit à 1).
- Le programmeur peut écrire directement dans le registre de direction pour mettre les lignes en entrée ou en sortie.
- Il faut bien sûr connaitre la correspondance entre les lignes du module Uno et les lignes des PORT du microcontrôleur.

La face cachée des fonctions digitalWrite et digitalRead

Nous venons de voir comment la fonction pinMode gère les E/S du module Arduino Uno pour les positionner soit en entrée, soit en sortie. Elle utilise pour cela le registre de contrôle DDR propre à chaque port d'entrée-sortie du microcontrôleur, ce que le programmeur peut faire de lui-même s'il cherche la rapidité lorsqu'il doit configurer plusieurs E/S. Maintenant que nous savons configurer nos E/S, il est temps d'apprendre à les utiliser.

digitalWrite

Une E/S configurée en sortie permet de commander un périphérique ; il suffit pour cela de mettre la ligne à l'état haut ou bien à l'état bas, en fonction de la façon dont le périphérique est raccordé.

Lorsque l'IDE trouve la fonction digitalWrite, il va tout simplement positionner à 0 ou à 1 le bit correspondant à la broche (pin) dans le registre PORT concerné. Par exemple, la broche 13 est reliée à la ligne 5 du PORTB, le compilateur positionnera donc à 1 (si HIGH, ou 0 si LOW) le 5ème bit du PORT B.

Le programmeur peut aussi le faire lui-même puisque le compilateur connait le nom des registres du microcontrôleur. Voyons cela sur un exemple, celui du programme Blink.

Ouvrir le programme Blink donné en exemple dans l'IDE d'Arduino. La fonction setup positionne bien notre ligne 13 du module Uno en sortie ; cette ligne correspond au 5ème bit du PORTB. Remplaçons dans la fonction loop la ligne d'instruction digitalWrite (13, HIGH) ; par PORTB = 0b00100000 ; et la ligne d'instruction digitalWrite (13, LOW) ; par PORTB = 0b00000000 ;

Le programme Blink fonctionne de la même façon.

On aurait pu aussi écrire PORTB = 32 ; (pour mettre à l'état HIGH) et PORTB = 0 ; (pour mettre à l'état LOW). En effet, 0b00100000 en binaire vaut 32 en décimal et 0b00000000 en binaire vaut 0 en décimal.

Il ne faut pas perdre de vue qu'un programme est **plus facile à relire s'il utilise des fonctions du « langage » d'Arduino** (dont la syntaxe est explicite) plutôt que s'il fait appel à des registres qui doivent être connus des programmeurs. Mais parfois, on gagne en rapidité et en place mémoire en manipulant directement les registres du microcontrôleur.

Ouvrez le programme Blink et effectuez une vérification du programme (avec l'icône vérifier) ; le programme utilise 928 octets (IDE version 1.6.12). Corrigez la fonction setup et la fonction loop en utilisant directement les registres DDRB et PORTB comme vous avez appris à le faire (DDRB = 32, PORTB = 32 ou 0). Vérifiez le programme à nouveau ; le programme n'utilise plus que 648 octets.

digitalRead

Une E/S configurée en entrée peut être lue par le microcontrôleur pour déterminer si la tension présente sur l'entrée est au niveau haut (HIGH ou 5 V) ou au niveau bas (LOW ou 0 V). L'ordre à utiliser est la fonction :

digitalRead(pin);

Ne pas oublier le point-virgule à la fin de la ligne d'instruction ni la majuscule à Read. De plus, vous pouvez, pour plus de lisibilité, mettre des espaces car ils sont ignorés par le compilateur (le programme qui transforme ce que vous avez écrit dans le langage machine du microcontrôleur ATmega328P). On peut donc écrire :

digitalRead (pin) ;

pin est le numéro de la broche du module Uno à lire.

Chaque ligne d'un port peut être lue à travers le registre de contrôle PIN associé (PINB, PINC, PIND).

Si on veut activer les résistances de pull-up sans utiliser la fonction pinMode, il faut configurer à la fois le registre DDR et le registre PORT du port concerné. Ceci est résumé par le tableau ci-dessous :

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

À retenir des fonctions digitalWrite et digitalRead :

- On peut positionner une sortie numérique à HIGH ou à LOW avec la fonction digitalWrite.
- On peut positionner une sortie numérique à HIGH ou à LOW en écrivant directement dans le registre PORT concerné.
- Un programme est plus facile à comprendre s'il est écrit avec les fonctions du « langage » d'Arduino plutôt que s'il est écrit avec les différents registres de contrôle.
- L'utilisation des registres de contrôle permet un gain en nombre d'octets de programme et une exécution plus rapide.
- On peut lire une entrée numérique avec la fonction digitalRead qui retourne HIGH ou LOW.
- Chaque ligne d'un port peut être lue avec le registre de contrôle PIN du port associé.
- Le montage d'une entrée peut faire appel à des résistances de pull-up ou de pull-down.
- Les microcontrôleurs sont pourvus de résistances de pull-up (et parfois de pull-down) qu'on peut activer par programmation.
- Pour activer les résistances de pull-up de l'ATmega328P, on peut utiliser la fonction pinMode en mode INPUT_PULLUP.
- On peut activer les résistances de pull-up de l'ATmega328P en configurant à 0 le registre DDR et à 1 le registre PORT.