ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

ENTRÉES-SORTIES

PIERRE-YVES ROCHAT, EPFL

RÉV 2016/02/08

CHAMPS DE BITS

Les ports d'entrée-sortie des microcontrôleurs sont le plus souvent vus par l'application comme des bits séparés, alors qu'ils sont physiquement adressés par groupe de 8 bits. Il faut donc disposer des outils nécessaires pour manipuler séparément des bits à l'intérieur d'un champs de bits (bit set).

Trois problèmes se posent : * mettre un ou plusieurs bits à la valeur 1 (set bit) * mettre un ou plusieurs bits à la valeur 0 (clear bit) * tester la valeur d'un bit (test bit).

Prenons un exemple concret : les différents bits de P1OUT (port d'entrée sortie 1) d'un MSP430 sont utilisés à diverses fins, certains comme entrées, d'autres comme sorties. Sur la broche P1.6 se trouve une LED, qu'on souhaite allumer ou éteindre à certains moments. Sur les broches P1.2 et P1.3 se trouvent des boutons-poussoirs.

SET BIT

Pour mettre un bit à la valeur 1, le problème se pose de la manière suivante: P1OUT contient les valeurs: x7 x6 x5 x4 x3 x2 x1 x0, toutes inconnues à priori. Après l'opération Set Bit sur le bit P1.6, on souhaite obtenir les valeurs suivantes: x7 1 x5 x4 x3 x2 x1 x0 dans P1OUT. Les lois de l'algèbre de Boole nous affirment les égalités suivantes: A . 0 = 0 A . 1 = A (1 est l'élément neutre du ET) A + 0 = A (0 est l'élément neutre du OU) A + 1 = 1 On remarque rapidement que l'opération OU logique va nous permettre de réaliser la mise à 1 d'un bit : P1OUT x7 x6 x5 x4 x3 x2 x1 x0 Second opérande 0 1 0 0 0 0 0 0 ---------- OU Résultat x7 1 x5 x4 x3 x2 x1 x0 L'opérateur OU s'appliquant à un champ de bits

s'écrit | en C. Il ne faut pas le confondre avec l'opérateur || qui s'applique à deux valeurs vues comme des booléans (la valeur 0 correspondant à faux et toute valeur différente de zéro correspondant à vrai). L'opération Set Bit s'écrit donc, dans notre exemple: P1OUT = P1OUT | 0x40; La syntaxe suivante est équivalente, mais plus compacte à écrire : P1OUT|= 0x40; Noter les valeurs directement en hexa-décimal, ou encore en décimal, rend les programmes peu lisibles. On préfèrera la syntaxe suivante:

```
P1OUT|= (1<<6); // ou P1OUT|= BIT6;
```

La constante BIT6 vaut *(1<<6)* dans les déclarations standard proposées pour les MSP430. C'est le rang du bit dans l'octet, ou autrement dit la puissance de 2 correspondante. L'opérateur de décalage est utilisé ici pour mettre le bit à sa place. Remarque importante: l'expression *(1<<6)* est évaluée à la compilation et non à l'exécution, vu qu'elle ne comporte que des constantes. Choisir d'écrire de manière lisible ne pénalise donc pas les performances du programme, ni la taille du binaire ! Clear bit De la même manière, on utilisera l'opération logique ET pour la mise à 0 d'un bit. Mais l'élément neutre est alors le 1 :

```
P1OUT  x7  x6  x5  x4  x3  x2  x1  x0
Second opérande    1   0   1   1   1   1   1   1
------------------------------  ET Résultat   x7  0   x5  x4  x3  x2  x1  x0
```

D'où l'expression: *P1OUT = P1OUT & 0xBF;*

On préfèrera la notation suivante : *P1OUT &=~(1<<6); // ou P1OUT &=~(BIT6);*

Rappel : l'opérateur ~ effectue une inversion bit-à-bit sur un champ de bits.

TEST BIT

L'utilisation d'un bouton-poussoir doit permettre d'effectuer un débranchement dans le cours du programme: si le bouton est pressé, alors telle action doit être réalisée. C'est la structure *if (condition) ...* du C.

Une condition est simplement représentée par un nombre: la condition est fausse si le nombre vaut 0, et vraie dans le cas contraire.

La lecture des valeurs se trouvant sur les broches de P1.0 à P1.7 se fait en lisant la valeur de *P1IN*. On cherche une opération logique dont le résultat sera 0 si le bit testé vaut 0 (condition fausse), alors que il sera non-nul si le bit testé vaut 1 (condition vraie). C'est la fonction ET qui va être utilisée:

```
P1IN   x7   x6   x5   x4   x3   x2   x1   x0
Second opérande    0    0    0    0    0    1    0    0
------------------------------------  ET
Résultat   0    0    0    0    0    x2   0    0
```

En C, on écrit : `if ( P1IN & (1<<2) )...  // ou if  ( P1IN & (BIT2) )`

La valeur binaire contenant le bit qu'on souhaite tester s'appelle un masque. En effet, l'opération ET entre un champ de bits et cette valeur permet de maquer les bits qui ne nous intéressent pas, afin de ne garder que le bit, ou les bits, à tester.

Exemple

Une pression sur les boutons-poussoirs ON et OFF doivent respectivement allumer et éteindre la LED. Le schéma n'indique que les ajouts par rapport au schéma de base, qui contient aussi les alimentation et les signaux de programmation.

Voici le programme correspondant :

```
#include <MSP430G2553.h>
int main() {
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR|= (1<<6); // LED en sortie
    P1OUT|= (1<<2) | (1<<3); // résistances en pull-up
    P1REN|= (1<<2)|(1<<3); // connexion des résistances sur les entrées
    while (1) { // boucle infinie
      if (!(P1IN & (1<<2)) P1OUT |= (1<<6); // bouton ON
      if (!(P1IN & (1<<3)) P1OUT &= ~(1<<6); // bouton OFF
    }
}
```