YNOV M1

Allumer les LED grâce au bouton User

Méthodes du polling et de l'interruption

Jean-Gabriel Massicot
13/12/2018

Table des matières

Rapport de TP : Allumer les LED grâce au bouton User

Méthode par polling

But

Le but de cette partie est d'allumer les LED successivement en utilisant la méthode du polling. Cette méthode consiste en une boucle qui vérifie en permanence une condition. Notre code va vérifier l'état du bouton. A chaque appui sur le bouton, la LED allumée change en suivant l'ordre des couleurs suivant : Vert, Rouge, Orange, Bleu, Vert…

Configuration des ports

Nous aurons besoin du port contenant les 4 LED de la carte, (GPIOD) ainsi que de celui du bouton User (GPIOA). On utilisera le fichier « stm32f4xx_hal.h » car il contient des fonctions qui agissent directement sur les registres, ce qui est plus facile à utiliser que les registres directement. Cela simplifie donc la configuration et la rend plus lisible.

Configuration des LED

Le port correspondant aux LED est le port GPIOD. Il faut tout d'abord activer l'horloge avec la commande :

```
__HAL_RCC_GPIOD_CLK_ENABLE();
```

En étudiant la documentation, on voit que cette commande agit sur le registre RCC_AHB1ENR et modifie le bit indiqué comme GPIODEN. Ce registre active l'horloge du port D.

6.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHS EN	ETHM ACPTP EN	ETHM ACRXE N	ETHM ACTXE N	ETHMA CEN	Res.	**DMA2D EN**	DMA2E N	DMA1E N	CCMDAT ARAMEN	Res.	BKPSR AMEN	Reserved	
	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCE N	Res.	GPIOK EN	GPIOJ EN	GPIOIE N	GPIOH EN	GPIOG EN	GPIOFE N	GPIOEEN	GPIOD EN	GPIOC EN	GPIO BEN	GPIO AEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 3 **GPIODEN:** IO port D clock enable
This bit is set and cleared by software.
0: IO port D clock disabled
1: IO port D clock enabled

Ensuite on configure les 4 pins correspondants aux LED. On utilise la commande :

```
GPIO_InitTypeDef leds;
```

On déclare donc un élément de type « GPIO_InitTypeDef » qui est un struct comportant des paramètres modifiables. Puis on déclare les pins (dont on a relevé le numéro dans la documentation) avec la commande :

```
leds.Pin = GPIO_PIN_12 | GPIO_PIN_13 |GPIO_PIN_14 |GPIO_PIN_15;
```

Puis on les déclare en sortie push/pull (mode actif sur 1) avec la commande :

```
leds.Mode = GPIO_MODE_OUTPUT_PP;
```

Avant d'envoyer la commande dans les registres avec la commande :

```
HAL_GPIO_Init(GPIOD, &leds);
```

Configuration du bouton

Le bouton User est sur le port GPIOA. Il se configure sensiblement de la même façon que les LED. Les commandes sont les suivantes :

```
__HAL_RCC_GPIOA_CLK_ENABLE();
GPIO_InitTypeDef bouton;
bouton.Pin = GPIO_PIN_0;
bouton.Mode = GPIO_MODE_IT_FALLING;
HAL_GPIO_Init(GPIOA, &bouton);
```

On active l'horloge du port D dans le registre RCC_AHB1ENR :

Bit 0 **GPIOAEN:** IO port A clock enable
This bit is set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled

On déclare le bouton, on définit son pin comme étant le 0 comme indiqué dans la documentation. On le met en mode falling, donc actif sur front descendant. Enfin on envoie les informations dans les registres.

Programme principal

Le programme principal comprend une boucle infinie. On y met une variable « etatbouton » dans laquelle on met l'état du bouton. On met créé aussi un compteur « counter » qui va servir à sélectionner les cas d'utilisation. On appelle la fonction de configuration. Puis on teste l'état du bouton avec un if, et on modifie le compteur. Enfin on prévoit les cas successifs. Le code est donc le suivant :

```
int main (void){

	int etatbouton;
	int counter = 0;
	config_pins();
	while (1){
		etatbouton = HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_0);
		if (etatbouton){
			counter += 1; //compte les leds
			switch (counter){
				case 1 :
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_12, 1);
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_15, 0);
					break;
				case 2 :
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_13, 1);
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_12, 0);
					break;
				case 3 :
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_14, 1);
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_13, 0);
					break;
				case 4 :
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_15, 1);
					HAL_GPIO_WritePin (GPIOD, GPIO_PIN_14, 0);
					counter = 0;
					break;

			}
			while (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_0)); //bloque jusqu'au
relachement du bouton (en cas de rebonds)
		}
	}
}
```

Méthode par Interruptions

But

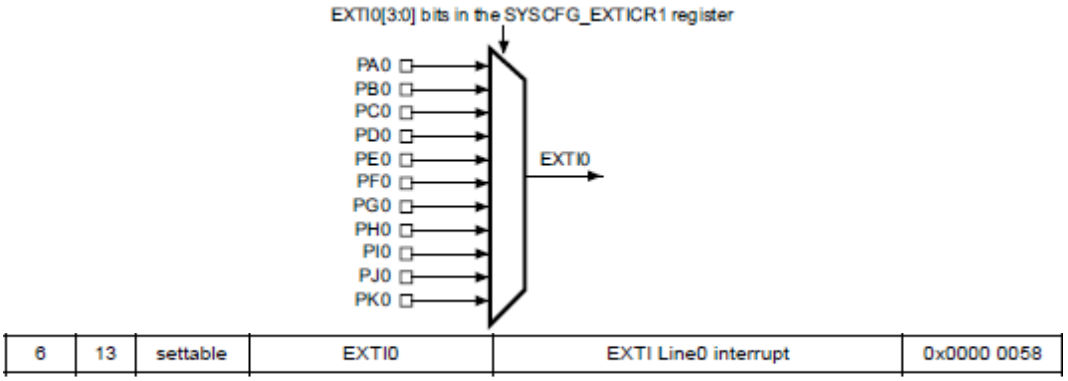
L'objectif est d'allumer les LED dans l'ordre : Rouge, Orange, Verte, Bleue. Le changement se fait sur relâchement du bouton au moyen d'une interruption.

Configuration des ports

Cette partie étant sensiblement la même, je développerai les éléments nouveaux et spécifiques par rapport à la première méthode, et donc développer en priorité les détails sur la configuration de l'interruption.

Configuration de l'interruption

Pour la configuration de l'interruption, on agit sur le registre NVIC (Nested Vectored Interrupt Controler). C'est l'élément qui va gérer le déclanchement des interruptions en fonction de leur priorité. Dans la datasheet, il faut savoir quel pin correspond à quelle interruption. Le bouton User est relié à PA0 qui lui-même est relié à EXTI0.



La position de l'interruption EXTI0 se trouve en 6ème position.
En sachant tout cela, nous pouvons configurer l'interruption sur le pin PA0. On défini la priorité de l'interruption en modifiant le registre IP dans NVIC. On utilise la commande suivante :

```
HAL_NVIC_SetPriority(EXTI0_IRQn, 5, 5);
```

On défini ainsi une priorité de 5 sur la 6ème position de l'interruption EXTI0. Cette priorité est en effet légèrement prioritaire sur le reste des instructions. Ensuite il faut activer l'interruption. EXTI0 permet de faire le lien avec le bouton qui fera réagir l'interruption. On utilise la commande suivante :

```
HAL_NVIC_EnableIRQ(EXTI0_IRQn);
```

Configuration du bouton

Pour configurer le bouton on utilise la commande suivante :

```
void EXTI0_IRQHandler(void){
	HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

Cette fonction de EXTI0 est prévue pour être paramétrée. On la définie en « lecture du bouton User » sur le GPIO pin 0.

En résumé l'appui sur le bouton déclenche EXTI0_IRQHandler. Cela créé une interruption. Ca appelle HAL_GPIO_EXTI_IRQHandler (GPIO_PIN_0) qui déclenche le callback en fonction du pin 0. Le callback est une fonction dans laquelle est traitée un événement précis.

Allumage des LED, fonction de callback

Cette fonction est déclarée ainsi :

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
```

Cette fonction avec les paramètres indiqués traite les évènements liés aux GPIO. La fonction traitant les cas d'allumage est dans cette fonction callback. Elle suit le même déroulement que dans la méthode précédente de polling :

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
	counter += 1; //compte les leds
		switch (counter){
			case 1 :
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_12, 1);
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_15, 0);
				break;
			case 2 :
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_13, 1);
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_12, 0);
				break;
			case 3 :
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_14, 1);
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_13, 0);
				break;
			case 4 :
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_15, 1);
				HAL_GPIO_WritePin (GPIOD, GPIO_PIN_14, 0);
				counter = 0;
				break;
		}
}
```

Fonction principale

La fonction principale est fortement réduite. Il ne reste en effet qu'à appeler les fonctions de configuration.

```
int main (void){
	config_pins();
	config_interrupt();

	while (1){
	}
}
```

Conclusion

Ce TP m'a permis de comprendre deux méthodes pour allumer des LED à partir du bouton User. Grâce à ces deux méthodes j'ai pu mieux prendre en main le logiciel Keil, et j'ai pu comprendre comment et où aller chercher les informations nécessaires. En comparant les deux méthodes, on peut dire que la méthode de l'interruption est plus efficace. En effet elle permet de ne pas tester l'état du bouton en permanence comme avec le polling. Elle ne va fonctionner qu'en cas d'appui sur le bouton, le reste du temps elle est inactive et libère donc le microcontrôleur et les ressources de la carte. Ainsi elle est un peu plus complexe à mettre en place mais elle fait fortement gagner en performance.