Axel Jacquot

M1 SER



Rapport d’exercice n°1 Led et Bouton

***Table des matières***

[Exercice n°1 2](#_Toc531216856)

[Enoncé : 2](#_Toc531216857)

[Explication des fonctions fournis pour la STM32 (HAL) : 2](#_Toc531216858)

[\_\_HAL\_RCC\_GPIO(AouD)\_CLK\_ENABLE.: 2](#_Toc531216859)

[GPIO\_InitTypeDef : 2](#_Toc531216860)

[HAL\_GPIO\_Init() : 4](#_Toc531216861)

[HAL\_GPIO\_WritePin() : 4](#_Toc531216862)

[HAL\_GPIO\_ReadPin() : 5](#_Toc531216863)

[Algorigramme : 6](#_Toc531216864)

[Programme Principal : 6](#_Toc531216865)

[Programme des Leds : 7](#_Toc531216866)

[Configuration des ports: 7](#_Toc531216867)

[Exercice n°2 8](#_Toc531216868)

[Enoncé : 8](#_Toc531216869)

[Explication du code fourni pour la STM32 : 8](#_Toc531216870)

[HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin): 8](#_Toc531216871)

[EXTI0\_IRQHandler : 8](#_Toc531216872)

[HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_0): 8](#_Toc531216873)

[HAL\_NVIC\_SetPriority(EXTI0\_IRQn, 5, 5): 8](#_Toc531216874)

[HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn): 9](#_Toc531216875)

[Algorigramme: 9](#_Toc531216876)

[Programme Principal: 9](#_Toc531216877)

[Interruption Led: 10](#_Toc531216878)

[Configuration des ports: 11](#_Toc531216879)

[Conclusion 12](#_Toc531216880)

# Exercice n°1

## Enoncé :

Lors de l’appui du bouton User se trouvant sur la carte STM32 nous devrons allumer les leds une à une tout en arrêtant la led qui on était allumé précédemment. Nous devrons allumer les Leds dans un ordre précis Vert, Rouge, Orange puis Bleu.

## Explication des fonctions fournis pour la STM32 (HAL) :

### \_\_HAL\_RCC\_GPIO(AouD)\_CLK\_ENABLE.:

Cette ligne de code permet d’activer les ports présents sur la carte STM32.

#define \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE() do { \

\_\_IO uint32\_t tmpreg = 0x00U; \

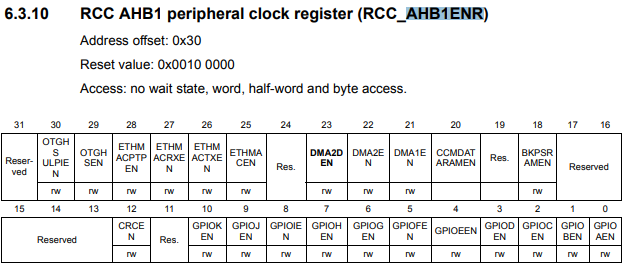
SET\_BIT(RCC->AHB1ENR, RCC\_AHB1ENR\_GPIODEN);\

/\* Delay after an RCC peripheral clock enabling \*/ \

tmpreg = READ\_BIT(RCC->AHB1ENR, RCC\_AHB1ENR\_GPIODEN);\

UNUSED(tmpreg); \

} while(0U)



Ce registre permet d’activer les ports A et D dans notre cas en mettant GPIODEN et GPIOAEN à 1.

### GPIO\_InitTypeDef :

typedef struct

{

uint32\_t Pin;

/\*!< Specifies the GPIO pins to be configured.

This parameter can be any value of @ref GPIO\_pins\_define \*/

uint32\_t Mode;

/\*!< Specifies the operating mode for the selected pins.

This parameter can be a value of @ref GPIO\_mode\_define \*/

uint32\_t Pull;

/\*!< Specifies the Pull-up or Pull-Down activation for the selected pins.

This parameter can be a value of @ref GPIO\_pull\_define \*/

uint32\_t Speed;

/\*!< Specifies the speed for the selected pins.

This parameter can be a value of @ref GPIO\_speed\_define \*/

uint32\_t Alternate;

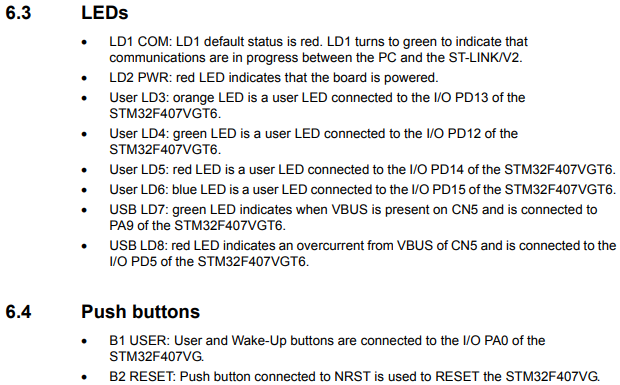
/\*!< Peripheral to be connected to the selected pins.

This parameter can be a value of @ref GPIO\_Alternate\_function\_selection \*/

}GPIO\_InitTypeDef;

Permet de régler les pins, le mode voulu des pins (entrée, sortie, interruption …), la vitesse, si les pins sont en pull-down ou en pull-up. Dans notre cas nous initialiserons les pins correspondant au led en sortie et le pin correspondant au bouton en entrée pour le 1er exercice et interruption pour le 2nd.

Les différents pins pour le bouton et les leds ci-dessous :



Pour les leds il s’agit des pins 12(vert),13(orange),14(rouge),15(bleu) du port D.

Pour le bouton il s’agit de la pin 0 du port A.

### HAL\_GPIO\_Init() :

Cette ligne permet d’initialiser les pins que nous avons régler avant sur le port que nous voulons. En paramètre de la fonction nous devons mettre le port et ensuite la configuration des différents pins.

### HAL\_GPIO\_WritePin() :

Change l’état des pins se trouvant sur le port qui lui sont mis en paramètre ainsi que l’état dans lequel nous voulons configurer le ou les pins.

void HAL\_GPIO\_WritePin(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin, GPIO\_PinState PinState)

{

//GPIOx prend la valeur d'un Port soit Port A ou D dans notre cas

//GPIO\_PIN correspond à ou aux pin avec lesquels nous voulons changer leurs états

//PinState est l'état dans lequel nous voulons mettre le pin

/\* Check the parameters \*/

assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));

assert\_param(IS\_GPIO\_PIN\_ACTION(PinState));

if(PinState != GPIO\_PIN\_RESET) //Vérifie si le pin doit être mis a 1

{

GPIOx->BSRR = GPIO\_Pin; //Le pin est mis à 1. BSRR permet d’agir sur l’état du pin

}

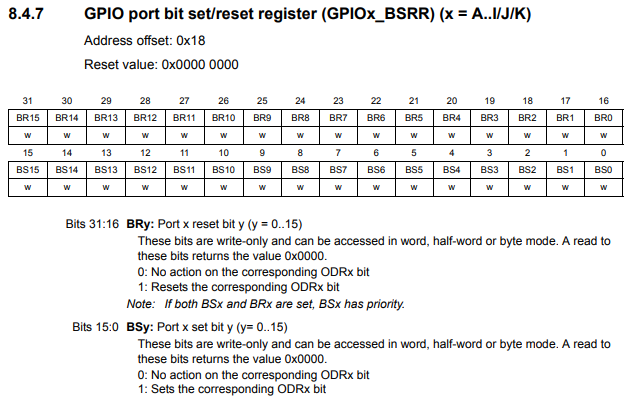
else

{

GPIOx->BSRR = (uint32\_t)GPIO\_Pin << 16U; //Le pin est mis à 0

}

}



### HAL\_GPIO\_ReadPin() :

Permet de lire l’état d’un pin se trouvant sur un port, nous devons mettre ceux-ci en paramètre de la fonction.

GPIO\_PinState HAL\_GPIO\_ReadPin(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

{

//GPIOx prend la valeur d'un Port soit Port A ou D dans notre cas

//GPIO\_PIN correspond a ou aux pin avec lesquels nous voulons changer leurs états

GPIO\_PinState bitstatus; //Déclaration d'un varibale qui récuperera l'état de lapin

/\* Check the parameters \*/

assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));

if((GPIOx->IDR & GPIO\_Pin) != (uint32\_t)GPIO\_PIN\_RESET)

//Vérifie si le pin n'est pas 0. IDR permet de lire les états des pins du port choisit

{

bitstatus = GPIO\_PIN\_SET; //variable de récupération mis à 1

}

else

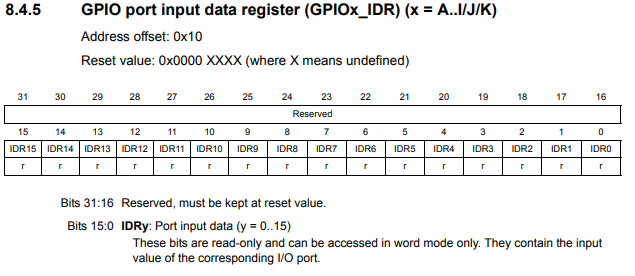
{

bitstatus = GPIO\_PIN\_RESET; //variable de récupération mis à 0

}

return bitstatus; //Sortie de la fonction avec en retour la variable de récupération

}



## Algorigramme :

### Programme Principal :



### Programme des Leds :



### Configuration des ports:



# Exercice n°2

## Enoncé :

Lors de l’appui du bouton User se trouvant sur la carte STM32 nous devrons déclencher une interruption qui aura pour but d’allumer les leds une à une tout en arrêtant la led qui on était allumé précédemment. Nous devrons allumer les Leds dans un ordre précis Rouge, Orange, Vert puis Bleu.

## Explication du code fourni pour la STM32 :

### HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin):

Cette fonction est l’emplacement ou nous allons mettre le code que nous souhaitons exécuter lors de l’interruption.

### EXTI0\_IRQHandler :

Fonction contenant la fonction permettant de vérifier si l’interruption doit être activé. EXTIO correspond à la première interruption GPIO qui est disponible.

### HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_0):

Fonction qui permet de lancer le code se trouvant dans l’interruption si celle-ci est demandé.

Le pin 0 est en paramètre de cette fonction pour spécifier à celle-ci sur quel pin du port réglé en interruption elle doit vérifier son état.

void HAL\_GPIO\_EXTI\_IRQHandler(uint16\_t GPIO\_Pin)

{

//GPIO\_PIN est le pin que nous souhaitons tester sur un port défini à l'avance dans notre cas le port A

/\* EXTI line interrupt detected \*/

if(\_\_HAL\_GPIO\_EXTI\_GET\_IT(GPIO\_Pin) != RESET) //Verifie si un pin du port A est à 1 pour pouvoir déckencher une interruption si c'est le cas

{

\_\_HAL\_GPIO\_EXTI\_CLEAR\_IT(GPIO\_Pin); //Nettoie l'interruption

HAL\_GPIO\_EXTI\_Callback(GPIO\_Pin); //Fonction permettant de lancer le code se trouvant dans l'interruption

}

}

### HAL\_NVIC\_SetPriority(EXTI0\_IRQn, 5, 5):

void HAL\_NVIC\_SetPriority(IRQn\_Type IRQn, uint32\_t PreemptPriority, uint32\_t SubPriority) //Dans STM32f4xx\_hal\_cortex.c

{

//IRQn\_Type IRQn Permet de choisir le type de l'interruption (GPIO,UART,I2C,SPI)

//PreemptPriority Permet de déterminer quand une interruption peut préempter une autre

//SubPriority permet de déterminer laquelle des interruptions de même priorité de préemption en attente s'exécutera en premier

uint32\_t prioritygroup = 0x00U;

/\* Check the parameters \*/

assert\_param(IS\_NVIC\_SUB\_PRIORITY(SubPriority));

assert\_param(IS\_NVIC\_PREEMPTION\_PRIORITY(PreemptPriority));

prioritygroup = NVIC\_GetPriorityGrouping();

NVIC\_SetPriority(IRQn, NVIC\_EncodePriority(prioritygroup, PreemptPriority, SubPriority));

//NVIC\_SetPriority permey de régler la priorité sur le type souhaité

}

### HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn):

Active l’interruption.

void HAL\_NVIC\_EnableIRQ(IRQn\_Type IRQn)

{

/\* Check the parameters \*/

assert\_param(IS\_NVIC\_DEVICE\_IRQ(IRQn));

/\* Enable interrupt \*/

NVIC\_EnableIRQ(IRQn);

}

## Algorigramme:

### Programme Principal:



### Interruption Led:



### Configuration des ports:



# Conclusion

Ces exercices nous on permit de plus approfondir les conséquences que nous avons acquises pendant les cours d’ARM et aussi de connaitre la déclaration des interruptions ainsi que leurs fonctions sur une carte STM32.