TD 5 STM32

Exercice 1. Trouver des adresses

- ✓ Donner l'adresse du périphérique RCC
- ✓ Donner l'adresse du périphérique Port A
- ✓ Donner l'adresse du périphérique Port C
- ✓ Donner l'adresse du périphérique SYS CFG
- ✓ Donner l'adresse du registre BSRR du port A
- ✓ Donner l'adresse du registre BRR du port C

Exercice 2. Masquage en C

Soit une variable var codée sur 32 bits. Ecrire en langage C, les fonctions actions suivantes

- ✓ Forcer bit 5, 7 et 13 à 1
- ✓ Forcer bit 4, 9 et 21 à 0
- ✓ Vérifier si bit 5 est à 1
- ✓ Basculer la valeur des bits 6 et 11

Exercice 3. Utilisation Pointeurs

Les exercices suivants

1) 0x4002 040C est l'adresse du registre GPIOx_PUPDR du port B. Comment écrire la valeur 0x00FFAA55 à cette adresse (on pourra utiliser int qui correspond à un entier 32-bit sur un Cortex m3)
2) Mettre l'adresse dans un define et réécrire la commande demandée dans la question 1
3) Avec la structure de GPIO, créer une variable GPIO_PortB comme pointeur sur une structure de type GPIO_TypeDef et initialiser le pointeur à l'adresse 0x40020400 qui est l'adresse du port B.
   Utiliser la structure pour réécrire la question 1

```
typedef struct
{
  __IO uint32_t MODER;        /*!< GPIO port mode register,                Address offset: 0x00      */
  __IO uint32_t OTYPER;       /*!< GPIO port output type register,         Address offset: 0x04      */
  __IO uint32_t OSPEEDR;      /*!< GPIO port output speed register,        Address offset: 0x08      */
  __IO uint32_t PUPDR;        /*!< GPIO port pull-up/pull-down register,   Address offset: 0x0C      */
  __IO uint32_t IDR;          /*!< GPIO port input data register,          Address offset: 0x10      */
  __IO uint32_t ODR;          /*!< GPIO port output data register,         Address offset: 0x14      */
  __IO uint32_t BSRR;         /*!< GPIO port bit set/reset registerBSRR,   Address offset: 0x18      */
  __IO uint32_t LCKR;         /*!< GPIO port configuration lock register,  Address offset: 0x1C      */
  __IO uint32_t AFR[2];       /*!< GPIO alternate function register,       Address offset: 0x20-0x24 */
  __IO uint32_t BRR;          /*!< GPIO bit reset register,                Address offset: 0x28      */
} GPIO_TypeDef;
```

Exercice 4. initialisation ports I/O / Allumage LED sur PA5

Réécrire la code ci-dessous écrit en assembleur en langage C (en utilisant la structure de GPIO).

```
typedef struct
{
  __IO uint32_t MODER;        /*!< GPIO port mode register,                    Address offset: 0x00      */
  __IO uint32_t OTYPER;       /*!< GPIO port output type register,             Address offset: 0x04      */
  __IO uint32_t OSPEEDR;      /*!< GPIO port output speed register,            Address offset: 0x08      */
  __IO uint32_t PUPDR;        /*!< GPIO port pull-up/pull-down register,       Address offset: 0x0C      */
  __IO uint32_t IDR;          /*!< GPIO port input data register,              Address offset: 0x10      */
  __IO uint32_t ODR;          /*!< GPIO port output data register,             Address offset: 0x14      */
  __IO uint32_t BSRR;         /*!< GPIO port bit set/reset registerBSRR,       Address offset: 0x18      */
  __IO uint32_t LCKR;         /*!< GPIO port configuration lock register,      Address offset: 0x1C      */
  __IO uint32_t AFR[2];       /*!< GPIO alternate function register,           Address offset: 0x20-0x24 */
  __IO uint32_t BRR;          /*!< GPIO bit reset register,                    Address offset: 0x28      */
} GPIO_TypeDef;
```

Code assembleur à réécrire en C

```
/* Enable port A */
ldr r0, =0x4002381C     /* address of the RCC_AHBENR GPIO Clock Enable Register */
ldr r1, =0x00000001     /* mask to apply to RCC_AHBENR to Enable GPIOAEN */
ldr r2, [r0]            /* load value of RCC_AHBENR to r2 */
orr r1,r2               /* for GPIOEN to 1 to enable port A */
str r1, [r0]            /* write new value to RCC_AHBENR */

/* configure pin in output mode */
ldr r0, =0x40020000     /* address of the MODE Register port A */
ldr r1, =0x00000400     /* mask to apply to MODE Register port A */
ldr r2, [r0]            /* load value of MODE Register port A  */
orr r1,r2               /* force bit 10 to 1  */
str r1, [r0]            /* write new value to port A MODER */

ldr r1, =0xFFFFF7FF     /* mask to apply to MODE Register port A */
ldr r2, [r0]            /* load value of MODE Register port A  */
and r1,r2               /* force bit 11 to 0  */
str r1, [r0]            /* write new value to port A MODE R */

ldr r0, =0x40020014     // address of port A ODR register
ldr r1, =0x00000020     /* mask to apply to port A ODR */
ldr r2, [r0]            /* load value of port A ODR */
orr r1,r2               /* for port A ODR to 1 to to put this pin to high level */
str r1, [r0]            /* write new value to port A ODR */
```