GPIO (GENERAL PURPOSE INPUT/OUTPUT)

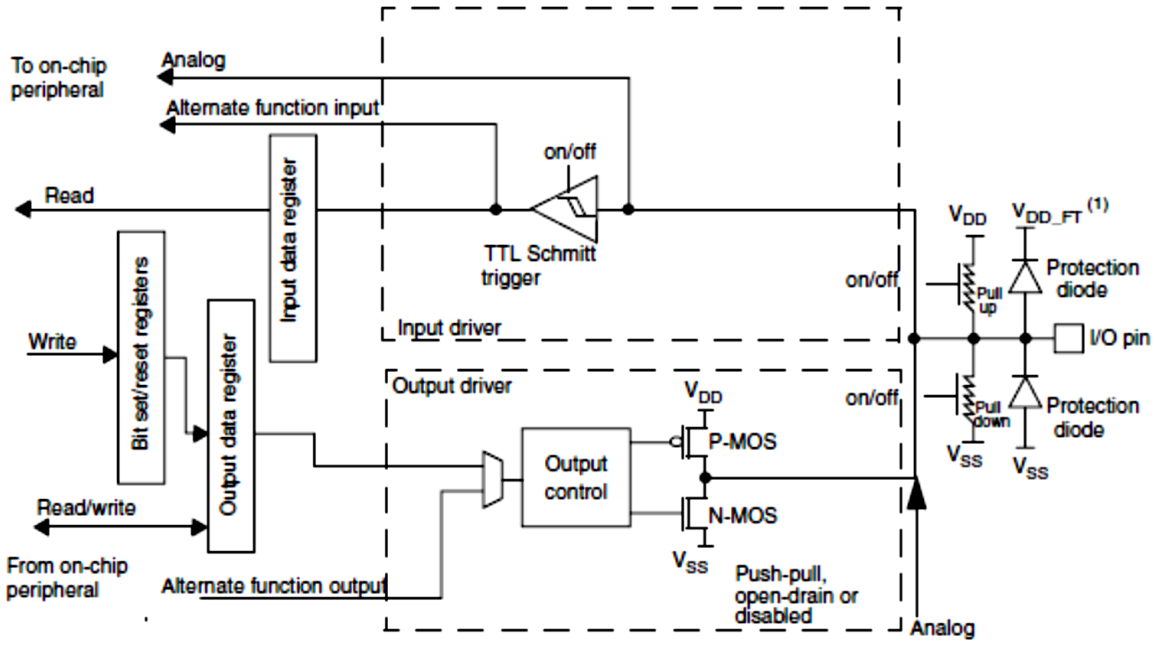
1. Présentation

STM32F407 dispose de 9 ports d'entrées/sorties de 16 bits (dénommés GPIOA à GPIOI), partagés avec d'autres périphériques,. A chaque port I/O est associé quatre registres de configuration 32 bits (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR et GPIOx_PUPDR), deux registres de données 32bits (GPIOx_IDR and GPIOx_ODR), un register de 32bits set/reset (GPIOx_BSRR), un registre 32bits de verrouillage (GPIOx_LCKR) et deux registres 32 bits pour les fonctions secondaire (Alternate function) (GPIOx_AFRH and GPIOx_AFRL).

2. Configuration des broches

Chaque broche du port I/O peut être configurée selon les modes suivants :

- Entrée flottante
- Entrée pull-up (PU)
- Entrée pull-down (PD)
- Analogique
- Sorite drain ouvert (OD) avec résistance pull-up ou pull-down
- Sortie push-pull (PP) avec résistance pull-up ou pull-down
- Alternate Function push-pull avec résistance pull-up ou pull-down
- Alternate Function drain ouvert avec résistance pull-up ou pull-down



La configuration de chaque proche est décrite par le tableau suivant :

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		Configuration des I/O	
01	0	SPEED [B:A]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reservé	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reservé (GP output OD)	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reservé	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reservé	
00	x	x	x	0	0	input	Floating
	x	x	x	0	1	input	PU
	x	x	x	1	0	input	PD
	x	x	x	1	1	Reservé (input floating)	
11	x	x	x	0	0	Input / output	Analog
	x	x	x	0	1	Reservé	
	x	x	x	1	0		
	x	x	x	1	1		

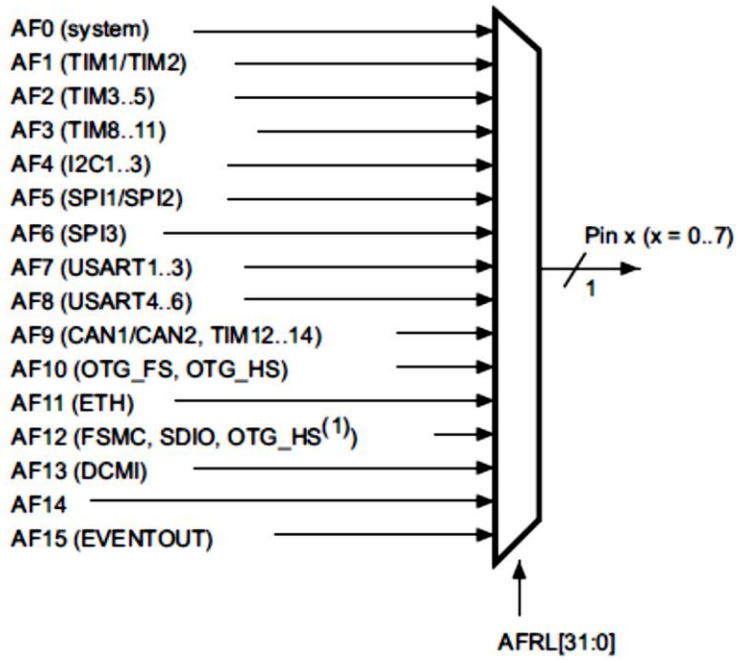
GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function

GPIO_Registers.pdf

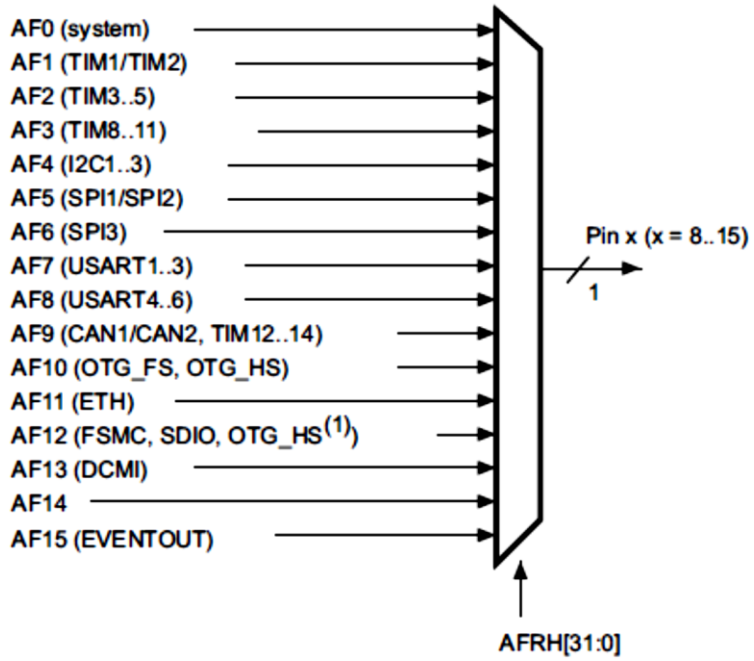
2.1. Multiplexage des proches

Chaque broche I/O peut être reliée à l'une de seize fonctions secondaires (Alternate Function AF0 .. AF15) à travers les registres GPIOx_AFRL et GPIO_AFRH :

For pins 0 to 7, the GPIOx_AFRL[31:0] register selects the dedicated alternate function



For pins 8 to 15, the GPIOx_AFRH[31:0] register selects the dedicated alternate function



pinsSTM32.pdf

3. Caractéristiques Electriques

Table 12. Current characteristics

Symbol	Ratings	Max.	Unit
I_{VDD}	Total current into V_{DD} power lines (source)(1)	240	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)(1)	240	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current source by any I/Os and control pin	25	
$I_{INJ(PIN)}$ (2)	Injected current on five-volt tolerant I/O(3)	–5/+0	
	Injected current on any other pin(4)	±5	
$\Sigma I_{INJ(PIN)}$ (4)	Total injected current (sum of all I/O and control pins)(5)	±25	

Table 48. I/O static characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IL}	FT, TTa and NRST I/O input low level voltage	1.7 V ≤V_{DD} ≤3.6 V	-	-	$0.3V_{DD}$-0.04(1)	V
			-	-	$0.3V_{DD}$(2)	
	BOOT0 I/O input low level voltage	1.75 V ≤V_{DD} ≤3.6 V -40 °C≤T_A ≤105 °C	-	-	$0.1V_{DD}$-+0.1(1)	
		1.7 V ≤V_{DD} ≤3.6 V 0 °C≤T_A ≤105 °C	-	-		
V_{IH}	FT, TTa and NRST I/O input low level voltage	1.7 V ≤V_{DD} ≤3.6 V	$0.45V_{DD}$+0.3(1)	-	-	
			$0.7V_{DD}$(2)	-	-	
	BOOT0 I/O input low level voltage	1.75 V ≤V_{DD} ≤3.6 V -40 °C≤T_A ≤105 °C	$0.17V_{DD}$+0.7(1)	-	-	
		1.7 V ≤V_{DD} ≤3.6 V 0 °C≤T_A ≤105 °C		-	-	

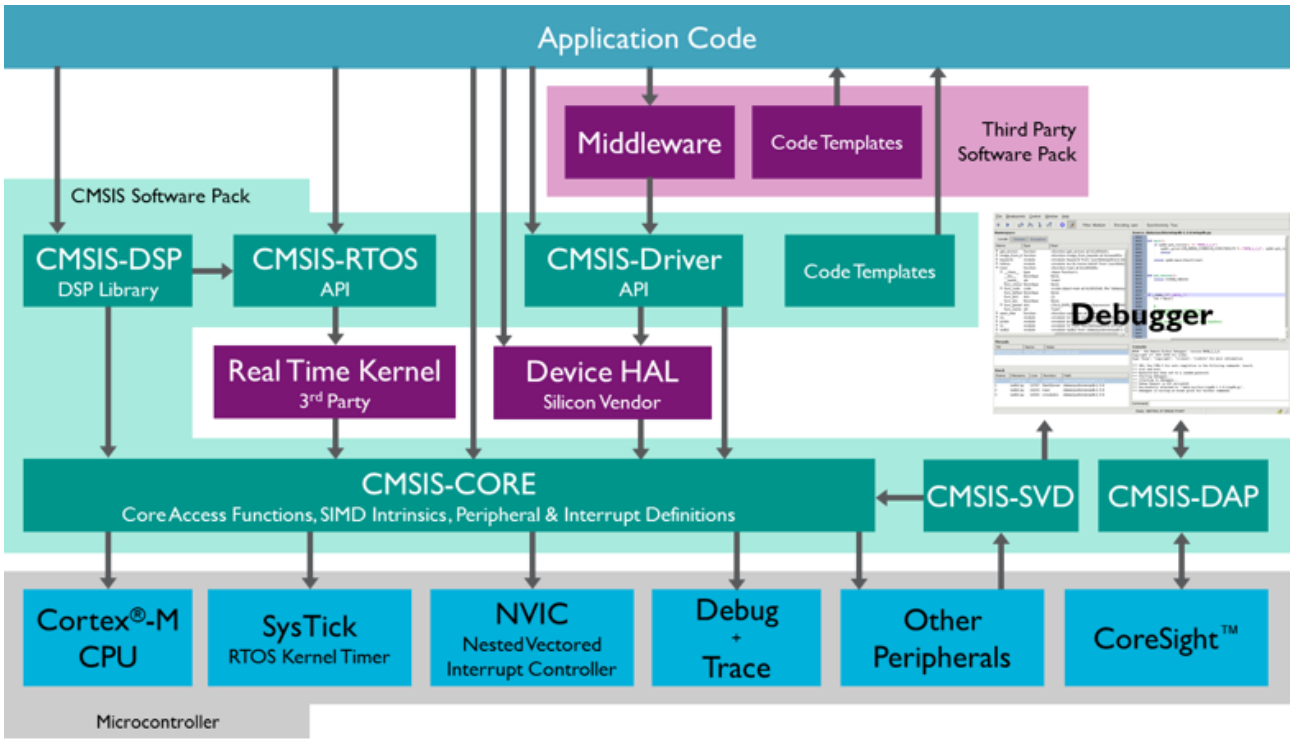
Table 49. Output voltage characteristics(1)

Symbol	Parameter	Conditions	Min	Max	Unit
V_{OL}(2)	Output low level voltage	CMOS port I_{IO} = +8 mA 2.7 V < V_{DD} < 3.6 V	-	0.4	V
V_{OH}(3)	Output high level voltage		V_{DD}–0.4	-	
V_{OL} (2)	Output low level voltage	TTL port I_{IO} =+ 8mA 2.7 V < V_{DD} < 3.6 V	-	0.4	V
V_{OH} (3)	Output high level voltage		2.4	-	
V_{OL}(2)(4)	Output low level voltage	I_{IO} = +20 mA 2.7 V < V_{DD} < 3.6 V	-	1.3	V
V_{OH}(3)(4)	Output high level voltage		V_{DD}–1.3	-	
V_{OL}(2)(4)	Output low level voltage	I_{IO} = +6 mA 2 V < V_{DD} < 2.7 V	-	0.4	V
V_{OH}(3)(4)	Output high level voltage		V_{DD}–0.4	-	

4. Librairie CMSIS : Cortex Microcontroller Software Interface Standard

Les fabricants des composants mettent à la disposition des développeurs une librairie « CMSIS software Pack ». La librairie « STM32 HAL Drivers », offre un

ensemble riche d'API pour interagir facilement avec les couches supérieures de l'application.



TYPE DES VARIABLES CMSIS

Type C Standard ANSI	Type MISRA C
`Signed char`	`int8_t`
`Signed short`	`int16_t`
`Signed int`	`int32_t`
`Signed __int64`	`int64_t`
`Unsigned char`	`uint8_t`
`Unsigned short`	`uint16_t`
`Unsigned int`	`uint32_t`
`Unsigned __int64`	`uint64_t`

Norme **MISRA** : Motor Industry Software Reliability Association

QUALIFICATEUR IO CMSIS

Qualificateur IO MISRA C	Type ANSI C	Description
`#define __I`	`Volatile const`	`Read only`
`#define __O`	`Volatile`	`Write only`
`#define __IO`	`Volatile`	`Read and write`

5. Manipulation des données en langage C (MapMemory.pdf)

5.1. Accès aux registres

L'adresse de base des périphériques : **`PERIPH_BASE = 0x40000000`**

GPIOA est situé au début de l'espace des périphériques, d'où :

```
#define GPIOA_BASE (PERIPH_BASE + 0x0000)
```

Les registres des ports d'entrées-sorties sont définis dans une structure :

```
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 uint16_t BSRRL;   /* GPIO port bit set/reset register,      Address offset: 0x18*/
  __IO uint32_t LCKR;    /* GPIO port configuration lock register,Address offset: 0x1C*/
  __IO uint32_t AFR[2]; /* GPIO alternate function registers,Address offset: 0x20-0x24*/
} GPIO_TypeDef;
```

On peut définir pour accéder aux éléments de la structure

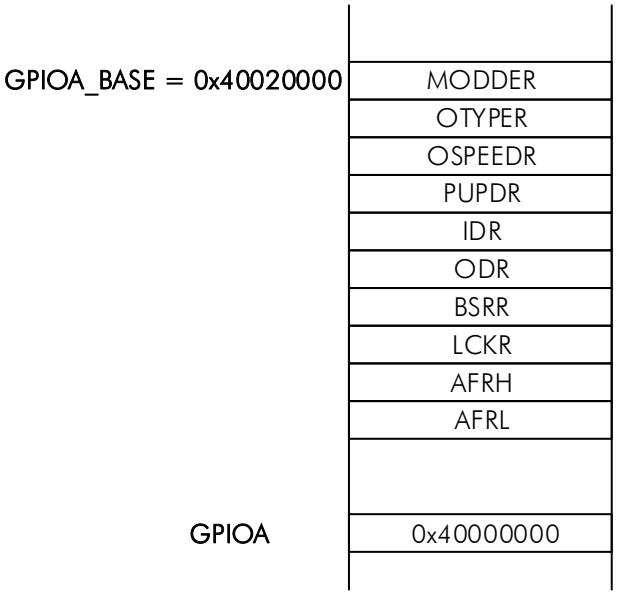
```
GPIO_TypeDef  *GPIOA ;
GPIOA = (GPIO_TypeDef *) GPIOA_BASE
```

Pour accéder aux registres :

```
(*GPIOA).ODR = *0x2FD0 ;
```

Ou mieux `GPIOA->ODR = *0x2FD0 ;`

Cette solution est gourmande en mémoire, on doit réserver une case mémoire pour chaque PORT.



On peut exploiter l'adresse directement sans passer par un pointeur :

```
((GPIO_TypeDef *) GPIOA_BASE)->ODR = 0x45ED ;
```

Il vaut mieux réduire l'écriture on définissant une équivalence :

```
#define  GPIOA   ((GPIO_TypeDef *) GPIOA_BASE)
```

Ce qui nous permet d'écrire :

```
GPIOA->ODR = *0x2FD0 ;
```

EXEMPLE1 :

1. Configurer les broches PD12, PD13, PD14 et PD15 en sortie.
2. Clignoter les 4 LEDs ensemble.

```
/*****************************************************************/
```

Initialisation :

```
RCC->AHB1ENR |= (1<<3);
GPIOD->MODER |= 0x55000000;
GPIOD->MODER &= 0x55FFFFFF;
GPIOD->OTYPER &= 0x0FFF;
GPIOD->OSPEEDR &=0x00FFFFFF;
```

Code :

Masquage

```
GPIOD->ODR &=0x0FFF;
HAL_Delay(500);
GPIOD->ODR |=0xF000;
HAL_Delay(500);
```

Bit SET/RESET

```
GPIOD->BSRR = (uint32_t)(0xF<<12);
HAL_Delay(500);
GPIOD->BSRR = (uint32_t)(0xF<<28);
HAL_Delay(500);
/************************************************/
```

5.2. Utilisation de la librairie CMSIS

Le fichier entête STM32F4xx.h comporte la définition des tous les registres des périphériques ainsi que la définition de toutes les constantes.

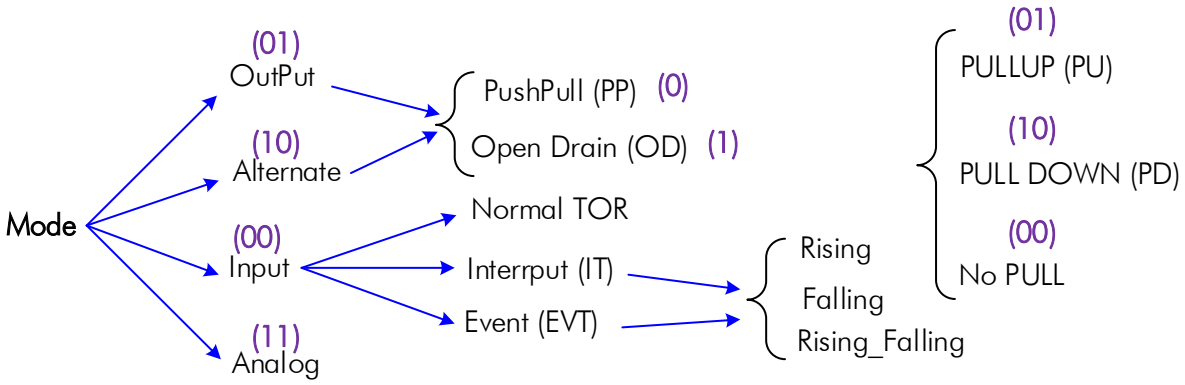
Définition des PORTS

```
#define APB1PERIPH_BASE       PERIPH_BASE

#define GPIOA_BASE            (AHB1PERIPH_BASE + 0x0000)
#define GPIOB_BASE            (AHB1PERIPH_BASE + 0x0400)
#define GPIOC_BASE            (AHB1PERIPH_BASE + 0x0800)
#define GPIOD_BASE            (AHB1PERIPH_BASE + 0x0C00)
#define GPIOE_BASE            (AHB1PERIPH_BASE + 0x1000)
#define GPIOF_BASE            (AHB1PERIPH_BASE + 0x1400)
```

```
#define GPIOG_BASE            (AHB1PERIPH_BASE + 0x1800)
#define GPIOH_BASE            (AHB1PERIPH_BASE + 0x1C00)
#define GPIOI_BASE            (AHB1PERIPH_BASE + 0x2000)


#define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF                ((GPIO_TypeDef *) GPIOF_BASE)
#define GPIOG                ((GPIO_TypeDef *) GPIOG_BASE)
#define GPIOH                ((GPIO_TypeDef *) GPIOH_BASE)
#define GPIOI                ((GPIO_TypeDef *) GPIOI_BASE)
```

CONFIGURATION DES BROCHES

Une broche peut se trouver d'un l'un des modes :



Structure de configuration d'une broche (fichier STM32F4xx_hal_gpio.h)

```
typedef struct
{
  uint32_t Pin;
  uint32_t Mode;
  uint32_t Pull;
  uint32_t Speed
  uint32_t Alternate;
}GPIO_InitTypeDef;
```

DEFINITION DU MODE

```
#define  GPIO_MODE_INPUT                        0x00000000U
#define  GPIO_MODE_OUTPUT_PP                    0x00000001U
#define  GPIO_MODE_OUTPUT_OD                    0x00000011U
#define  GPIO_MODE_AF_PP                        0x00000002U
#define  GPIO_MODE_AF_OD                        0x00000012U
#define  GPIO_MODE_ANALOG                       0x00000003U
#define  GPIO_MODE_IT_RISING                    0x10110000U
#define  GPIO_MODE_IT_FALLING                   0x10210000U
#define  GPIO_MODE_IT_RISING_FALLING            0x10310000U
```

```
#define  GPIO_MODE_EVT_RISING                    0x10120000U
#define  GPIO_MODE_EVT_FALLING                   0x10220000U
#define  GPIO_MODE_EVT_RISING_FALLING            0x10320000U
```

DEFINITION DE LA VITESSE

```
#define  GPIO_SPEED_FREQ_LOW         0x00000000U
#define  GPIO_SPEED_FREQ_MEDIUM      0x00000001U
#define  GPIO_SPEED_FREQ_HIGH        0x00000002U
#define  GPIO_SPEED_FREQ_VERY_HIGH   0x00000003U
```

DEFINITION DES RESISTANCES DE TIRAGE

```
#define  GPIO_NOPULL        0x00000000U
#define  GPIO_PULLUP        0x00000001U
#define  GPIO_PULLDOWN      0x00000002U
```

ENUMERATION BIT SET / RESER

```
 typedef enum
{
  GPIO_PIN_RESET = 0,
  GPIO_PIN_SET
}GPIO_PinState;
```

DEFINITION DES BROCHES IO

```
#define GPIO_PIN_0                 ((uint16_t)0x0001)
#define GPIO_PIN_1                 ((uint16_t)0x0002)
#define GPIO_PIN_2                 ((uint16_t)0x0004)
#define GPIO_PIN_3                 ((uint16_t)0x0008)
#define GPIO_PIN_4                 ((uint16_t)0x0010)
        ⋮                                ⋮
#define GPIO_PIN_15                ((uint16_t)0x8000)
#define GPIO_PIN_All               ((uint16_t)0xFFFF)
```

ALTERNATE PINS

Fichier « stm32f4xx_hal_gpio_ex.h »

FONCTIONS DE MANIPULATION DES IO

Reprenons l'exemple des LEDs connectées aux broches PD12, PD13, PD14 et PD15

1. Validation de l'horloge du GPIOD :
   appel à la fonction : `__HAL_RCC_GPIOD_CLK_ENABLE();`
2. Déclaration d'une structure de type : **`GPIO_InitTypeDef`**

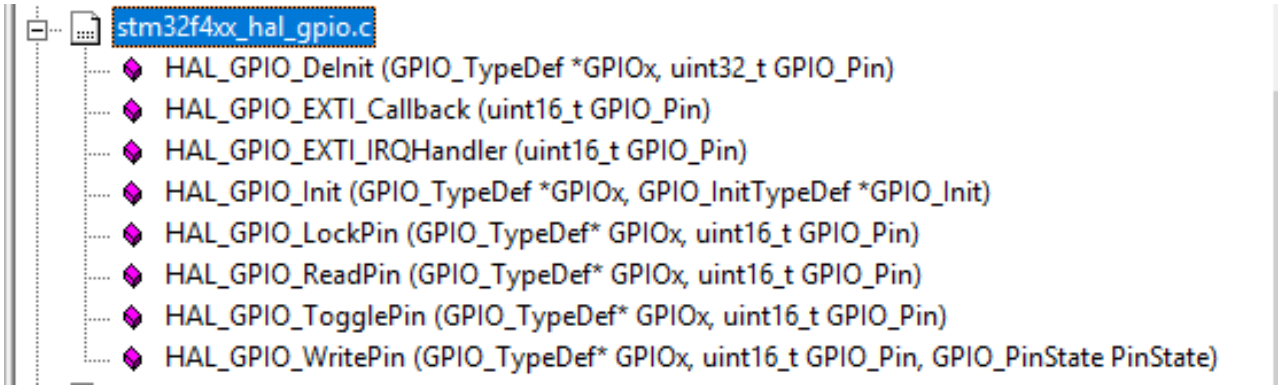
```
GPIO_InitTypeDef        GPIO_InitStruct ;
```

3. Initialisation des champs de la structure :

```
GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
```

4. Affectation des éléments de la structure aux registres de configurations du GPIOD

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



Exemple : `HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET) ;`

5.3. Généralisation

Pour faciliter l'utilisation de la librairie CMSIS, la procédure précédente s'applique à tous les périphériques : Timer, ADC, DAC, USART, SPI…

1. Les registres d'un périphérique donné sont définis dans une structure de type :

   `PPP_TypeDef` (PPP : non du périphérique, exemple `GPIO_TypeDef`)

2. Les bits de configuration du périphérique sont définis dans une structure de type :

   `PPP_InitTypeDef` (exemple : `GPIO_InitTypeDef`)