

Programmation des GPIOs par utilisation de la “ST Firmware Library”

DAMERGI Emir

OUTLINE

- Le Firmware.
- Présentation de la ST FWL « FirmWare Library ».
- Utilisation de la ST FWL
 - Structure d'un programme.
 - Activation des horloges des interfaces.
 - Configuration & Initialisation des interfaces E/S
 - Propriétés & Paramètres: application au GPIO
 - Fonctions relatives au GPIO.
- Pratique:
 - Exemple 1,2: clignotement Leds.
 - Exemple 3: Lecture état bouton poussoir + commande Leds
 - Création & configuration d'un projet IAR
 - Test de l'application sur carte

Approches de programmation des interfaces E/S

Il est possible de programmer les interfaces E/S par accès direct aux registres de ces derniers.



Il faut connaître plusieurs détails techniques:

- Les adresses des registre de l'interface E/S.
- La structure des registres et la signification de leurs contenus



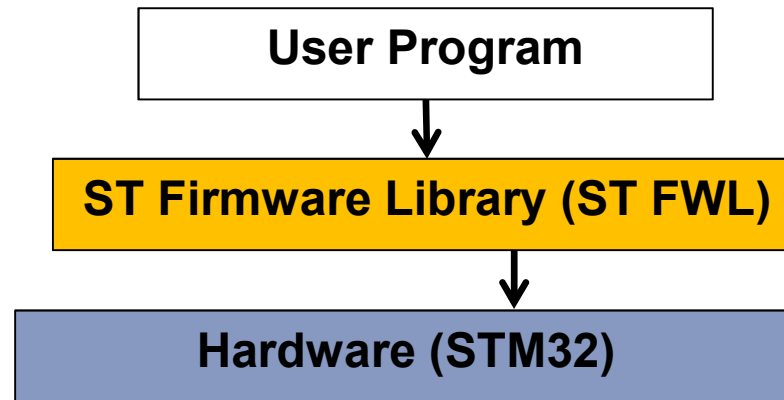
Approche compliquée et demande beaucoup de temps de développement

Approches de programmation des interfaces E/S

- Les constructeurs des microcontrôleurs (et des SOC's en général) fournissent des bibliothèques permettant d'exploiter les fonctionnalités des interfaces E/S: on parle de **FIRMWARE**.
- Le **Firmware**: du code (écrit en 'C' ou en langage Assembleur), donc il s'agit d'un **software** qui dépend complètement de la plateforme matérielle « **Hardware** » sur laquelle il est exécuté et ne peut être porté sur d'autres plateformes.
- Il est généralement développé par le constructeur même. D'où le terme **Firmware** (Firm = entreprise).

Approches de programmation des interfaces E/S

La Firmware Library de ST vient s'interposer entre le programme utilisateur et le Hardware.



Ceci offre à l'utilisateur une programmation **transparente** des Interfaces E/S : C'est-à-dire ça permet de faire **abstraction des détails techniques** relatifs au hardware (adresses et structures des registres)

Présentation de la ST FWL -1

Dans la suite on utilisera les notations suivantes:

- **PPP**: pour indiquer le type d'une interface E/S

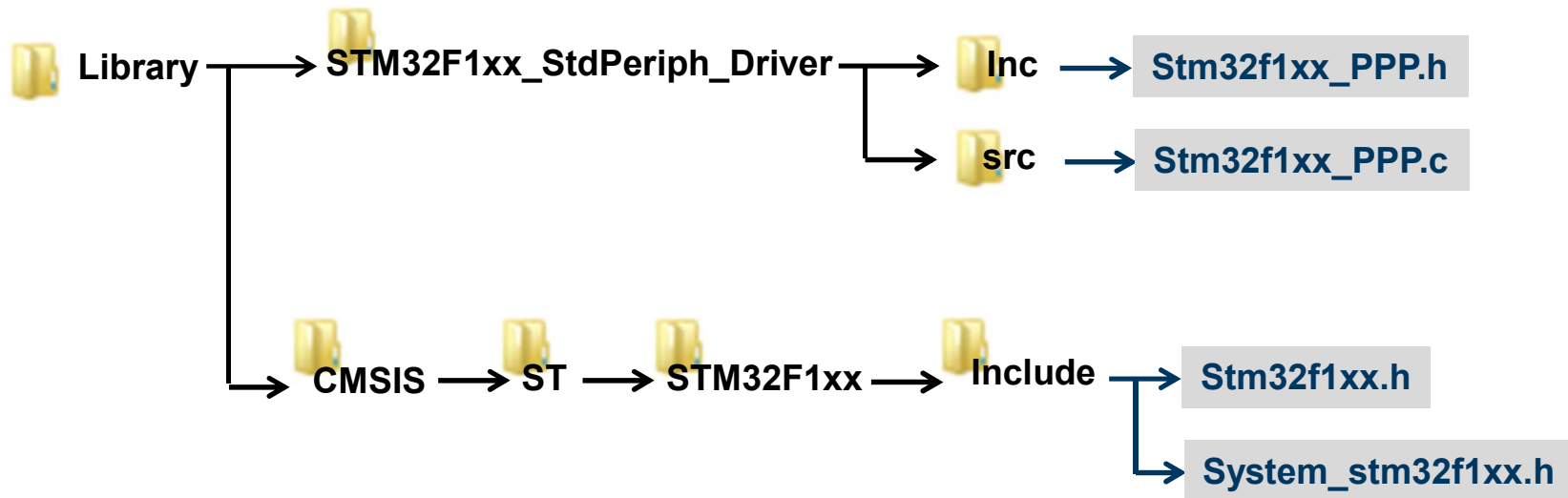
exemple: **GPIO,**
 USART,
 USB.

- **PPPx** pour indiquer une interface E/S précise

exemple: GPIO**A**, GPIO**B**,, GPIO**I**.
 USART**1**, USART**2**,, USART**6**.
 USB**1**, USB**2**.

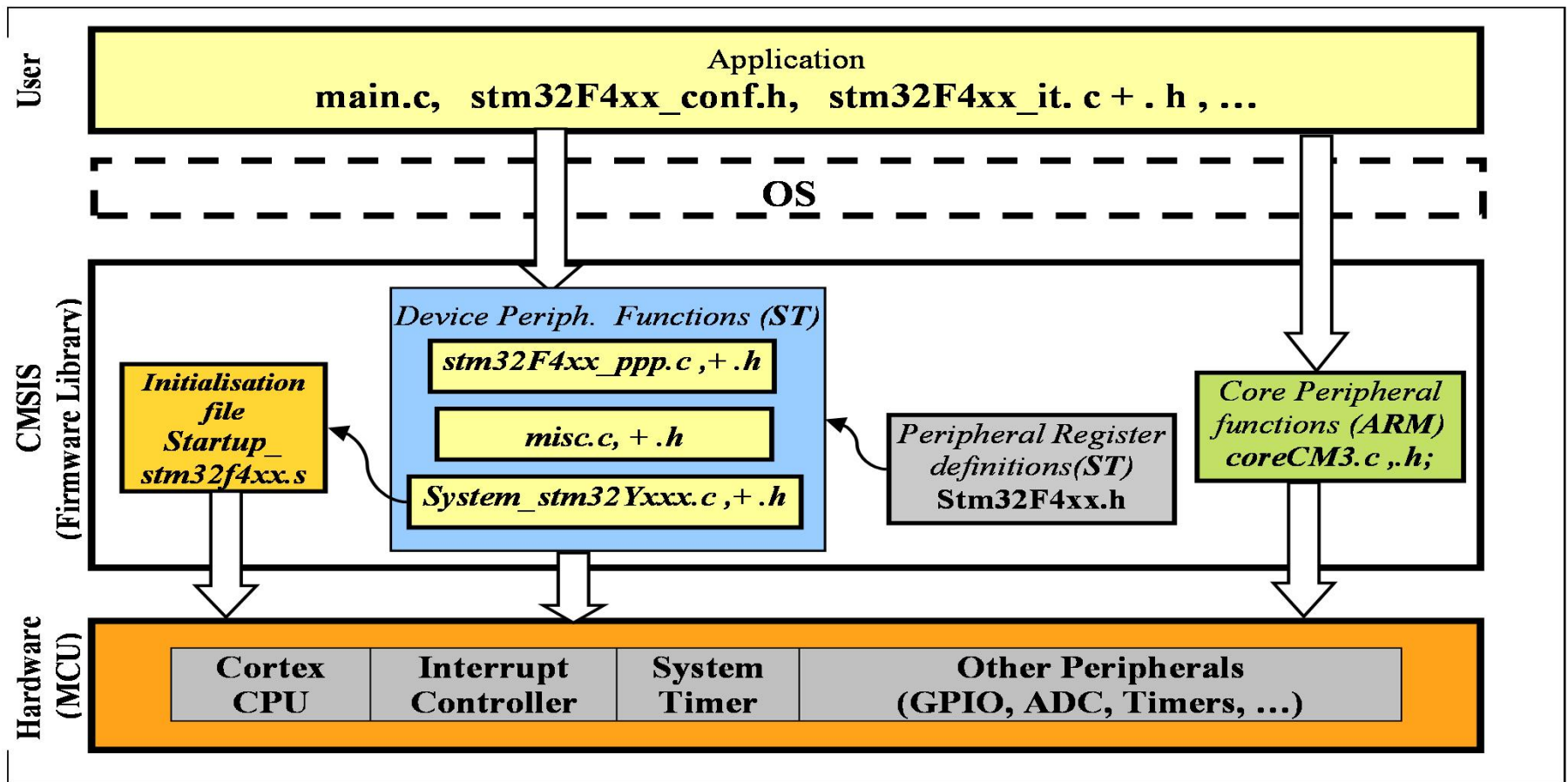
Présentation de la ST FWL- 2

La ST FWL est un ensemble de fichiers ayant l'arborescence suivante:



PPP = *GPIO, USART, ADC, DAC, TIM, USB, SPI, CRYP, etc...*

Présentation de la ST FWL-3

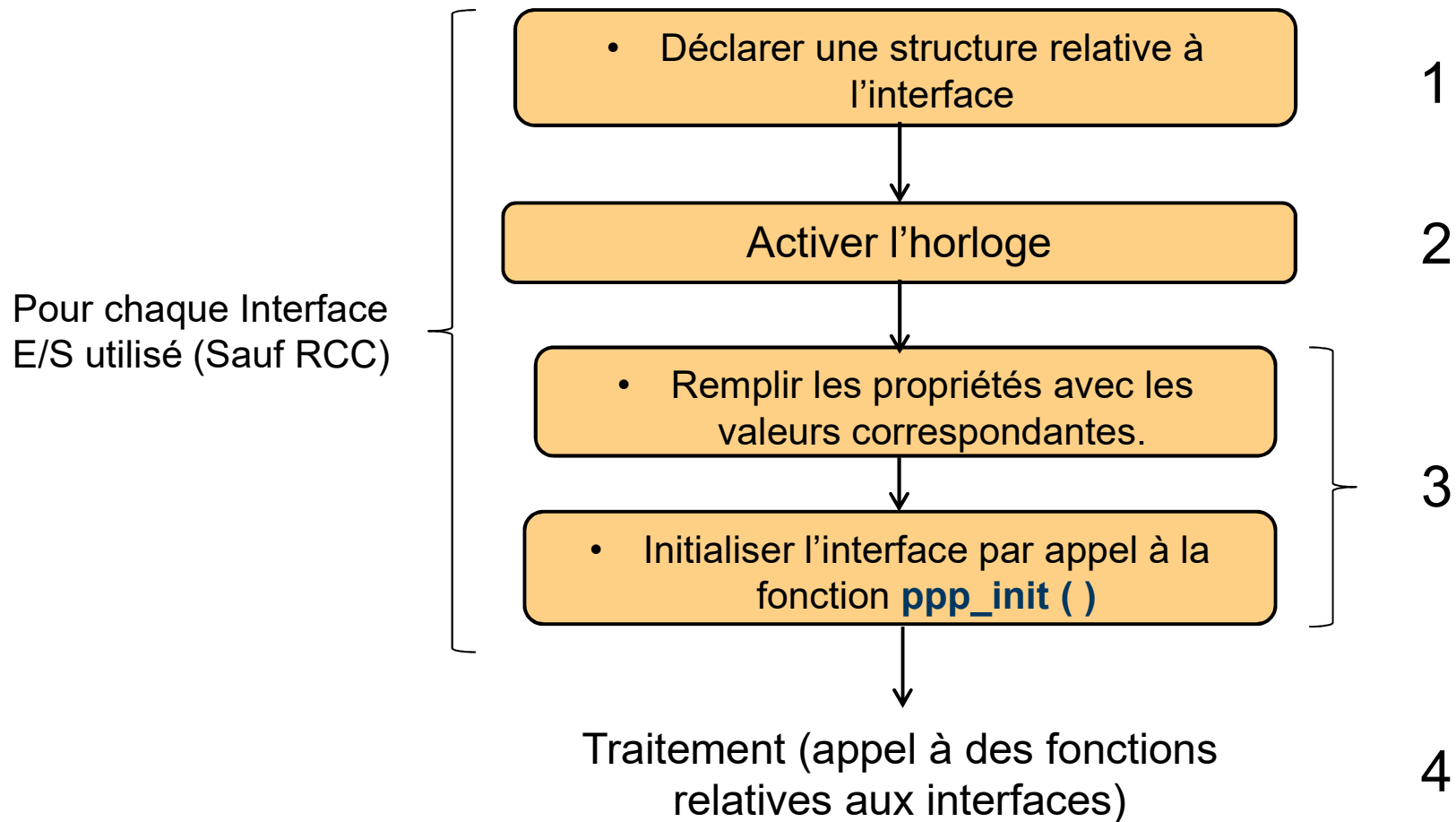


Présentation de la ST FWL-4

Fichier	Contenu
Stm32f1xx.h	<ul style="list-style-type: none">• Les adresses des interfaces E/S.• Une description des registres constituant les interfaces
Stm32f1xx_PPP.h <i>Exemple:</i> Stm32f4xx_gpio.h	<ul style="list-style-type: none">• Une description des paramètres de configuration de l'interface PPP.
Stm32f1xx_PPP.c <i>Exemple:</i> Stm32f4xx_gpio.c	<ul style="list-style-type: none">• Les fonctions permettant de manipuler les interfaces E/S PPP (<i>Initialisation, configuration , traitement</i>)

Utilisation de la librairie ST FWL -1

Tout programme utilisant les interfaces E/S possède la structure suivante:



Utilisation de la librairie ST FWL - 3

1 – Déclarer une structure :

PPP_InitTypeDef **PPP_InitStructure;**

Exemples:

- Pour l'interface GPIO (PPP = GPIO):

GPIO_InitTypeDef **GPIO_InitStructure;**

- Pour l'interface USART (PPP = USART):

USART_InitTypeDef **USART_InitStructure;**

Utilisation de la librairie ST FWL -2

2. Activer les horloges des interfaces E/S **PPPx**:

Ceci est effectué en faisant appel à la **fonction**


```
RCC_BUSxPeriphClockCmd ( RCC_BUSxPeriph_PPPx, ENABLE );
```

BUSx étant le bus auquel est connecté l'interface E/S à activer et pouvant être:

AHB1, AHB2, APB1 ou APB2

Exemple: pour activer **GPIOA** qui est connecté au bus **APB2**:

PPPx = GPIOA & **BUSx** = AHB1

 `RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOA, ENABLE);`

Utilisation de la librairie ST FWL - 3

3 - Configuration d'une interface E/S PPPx:

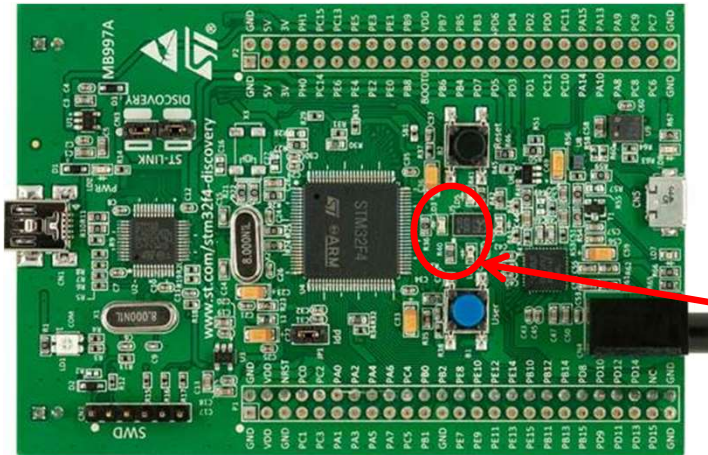
- A. Remplir les propriétés de la structure avec les valeurs appropriées:

```
PPP_InitStructure.propriété1 = valeur1;  
PPP_InitStructure.propriété2 = valeur2;  
.....;  
PPP_InitStructure.propriétéN = valeurN;
```

- B. Initialiser l'interface **PPPx** avec les valeurs déterminées.

```
PPP_Init (PPPx, &PPP_InitStructure);
```

Utilisation de la librairie ST FWL – 4 : Exemple avec GPIO



La carte Discovery F contient une Led:
connectée au **pin 9** du port **GPIOC**

Ecrire un programme permettant de faire clignoter la diode Led.

- *Les transistors devant être en push-pull.*
- *Pas de résistances (ni pull-up, ni pull-down).*

Utilisation de la librairie ST FWL – 5 : Exemple avec GPIO

1 – Déclarer une structure relative l'interface E/S PPPx:

PPP_InitTypeDef PPP_InitStructure;



GPIO_InitTypeDef GPIO_InitStructure;

Utilisation de la librairie ST FWL – 6 : Exemple avec GPIO

2. Activer des horloges des interfaces E/S **PPPx**:

```
RCC_BUSxPeriphClockCmd ( RCC_BUSxPeriph_PPPx, ENABLE );
```

Dans notre exemple, il s'agit d'activer l'interface
GPIOC qui est connecté au bus **APB2**.

 L'activation de l'horloge de GPIOC se fait avec:

```
RCC_APB2PeriphClockCmd ( RCC_APB2Periph_GPIOC, ENABLE );
```


Utilisation de la librairie ST FWL - 7

3 - Configuration d'une interface E/S PPPx - suite:

A. Remplir les propriétés de la structure avec les valeurs appropriées:

```
GPIO_InitStructure.propriété1 = valeur1;
```

```
.....;
```

```
GPIO_InitStructure.propriétéN = valeurN;
```

- **LES PROPRIÉTÉS ?**
- **LES VALUERS à affecter aux PROPRIÉTÉS ?**

Utilisation de la librairie ST FWL - 8

Ainsi on peut déclarer les propriétés de la structure ***GPIO_InitStructure*** :

```
GPIO_InitStructure.GPIO_Pin = .....;  
GPIO_InitStructure.GPIO_Mode = .....;  
GPIO_InitStructure.GPIO_Speed = .....;
```

Utilisation de la librairie ST FWL - 9

```
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

```
GPIO_Mode_AIN = 0x0,  
GPIO_Mode_IN_FLOATING = 0x04,  
GPIO_Mode_IPD = 0x28,  
GPIO_Mode_IPU = 0x48,  
GPIO_Mode_Out_OD = 0x14,  
GPIO_Mode_Out_PP = 0x10,  
GPIO_Mode_AF_OD = 0x1C,  
GPIO_Mode_AF_PP = 0x18
```

Utilisation de la librairie ST FWL - 10

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
```

```
GPIO_Speed_10MHz :  
GPIO_Speed_2MHz,  
GPIO_Speed_50MHz
```

Utilisation de la librairie ST FWL - 11

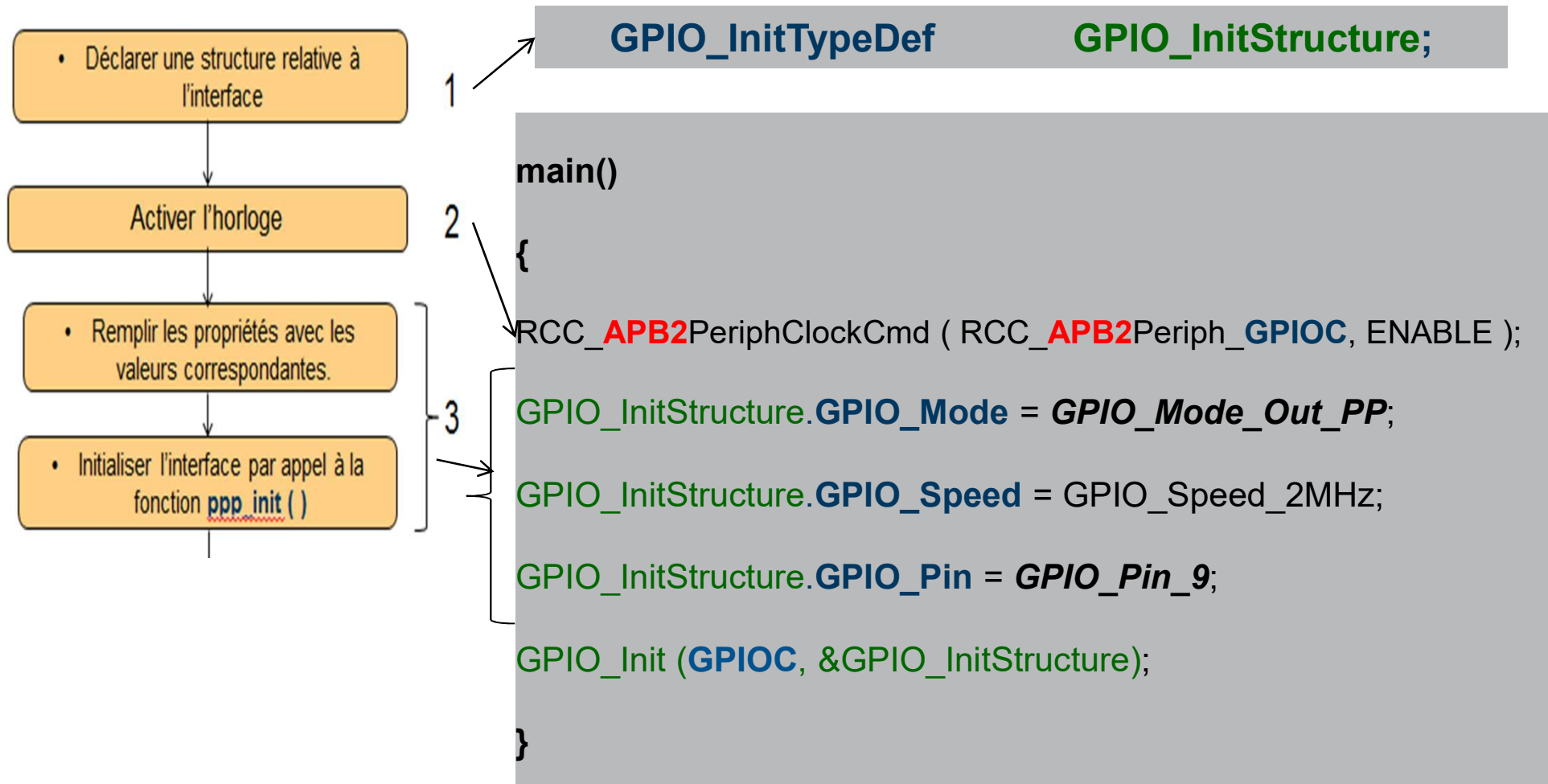
`GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;`

Defines

#define	<u>GPIO_Pin_0</u>	((uint16_t)0x0001)
#define	<u>GPIO_Pin_1</u>	((uint16_t)0x0002)
#define	<u>GPIO_Pin_10</u>	((uint16_t)0x0400)
#define	<u>GPIO_Pin_11</u>	((uint16_t)0x0800)
#define	<u>GPIO_Pin_12</u>	((uint16_t)0x1000)
#define	<u>GPIO_Pin_13</u>	((uint16_t)0x2000)
#define	<u>GPIO_Pin_14</u>	((uint16_t)0x4000)
#define	<u>GPIO_Pin_15</u>	((uint16_t)0x8000)
#define	<u>GPIO_Pin_2</u>	((uint16_t)0x0004)
#define	<u>GPIO_Pin_3</u>	((uint16_t)0x0008)
#define	<u>GPIO_Pin_4</u>	((uint16_t)0x0010)
#define	<u>GPIO_Pin_5</u>	((uint16_t)0x0020)
#define	<u>GPIO_Pin_6</u>	((uint16_t)0x0040)
#define	<u>GPIO_Pin_7</u>	((uint16_t)0x0080)
#define	<u>GPIO_Pin_8</u>	((uint16_t)0x0100)
#define	<u>GPIO_Pin_9</u>	((uint16_t)0x0200)
#define	<u>GPIO_Pin_All</u>	((uint16_t)0xFFFF)

Utilisation de la librairie ST FWL – 12 : Exemple avec GPIO

Led: **pin 9** du port **GPIOC**



Utilisation de la librairie ST FWL – 13: Fonctions

Functions

void	<u>GPIO_DeInit</u> (<u>GPIO_TypeDef</u> *GPIOx) De-initializes the GPIOx peripheral registers to their default reset values.
void	<u>GPIO_Init</u> (<u>GPIO_TypeDef</u> *GPIOx, <u>GPIO_InitTypeDef</u> *GPIO_InitStruct) Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.
void	<u>GPIO_PinAFConfig</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_PinSource, uint8_t GPIO_AF) Changes the mapping of the specified pin.
void	<u>GPIO_PinLockConfig</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_Pin) Locks GPIO Pins configuration registers.
uint16_t	<u>GPIO_ReadInputData</u> (<u>GPIO_TypeDef</u> *GPIOx) Reads the specified GPIO input data port.
uint8_t	<u>GPIO_ReadInputDataBit</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_Pin) Reads the specified input port pin.
uint16_t	<u>GPIO_ReadOutputData</u> (<u>GPIO_TypeDef</u> *GPIOx) Reads the specified GPIO output data port.
uint8_t	<u>GPIO_ReadOutputDataBit</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_Pin) Reads the specified output data port bit.
void	<u>GPIO_ResetBits</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_Pin) Clears the selected data port bits.
void	<u>GPIO_SetBits</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_Pin) Sets the selected data port bits.
void	<u>GPIO_StructInit</u> (<u>GPIO_InitTypeDef</u> *GPIO_InitStruct) Fills each GPIO_InitStruct member with its default value.
void	<u>GPIO_ToggleBits</u> (<u>GPIO_TypeDef</u> *GPIOx, uint16_t GPIO_Pin) Toggles the specified GPIO pins.

Utilisation de la librairie ST FWL - 14

```
/**
 * @brief Sets the selected data port bits.
 * @param GPIOx: where x can be (A..G) to select the GPIO peripheral.
 * @param GPIO_Pin: specifies the port bits to be written.
 * This parameter can be any combination of GPIO_Pin_x where x can be (0..15)
 * @retval None
 */
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
    /* Check the parameters */
    assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
    assert_param(IS_GPIO_PIN(GPIO_Pin));

    GPIOx->BSRR = GPIO_Pin;
}
```

Mettre à 1 le pin 9 du GPIOC:



`GPIO_SetBits (GPIOC, GPIO_Pin_9)`

Utilisation de la librairie ST FWL - 15

```
/**
 * @brief Clears the selected data port bits.
 * @param GPIOx: where x can be (A..G) to select the GPIO peripheral.
 * @param GPIO_Pin: specifies the port bits to be written.
 * This parameter can be any combination of GPIO_Pin_x where x can be (0..15)
 * @retval None
 */
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
    /* Check the parameters */
    assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
    assert_param(IS_GPIO_PIN(GPIO_Pin));

    GPIOx->BRR = GPIO_Pin;
}
```

Mettre à 0 le pin 9 du GPIOC:



GPIO_ResetBits (**GPIOC**, **GPIO_Pin_9**)

Utilisation de la librairie ST FWL - 16

Programme complet:

```
#include stm32f1xx.h
```

```
GPIO_InitTypeDef  GPIO_InitStructure;
```

```
Main () {
```

```
RCC_AHB1PeriphClockCmd ( RCC_APB2Periph_GPIOC, ENABLE );
```

```
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
```

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2Mhz;
```

```
GPIO_Init (GPIOC, &GPIO_InitStructure)
```

```
while (1) {
```

```
GPIO_SetBits (GPIOC, GPIO_Pin_9);
```

```
Delay();
```

```
GPIO_ResetBits (GPIOC, GPIO_Pin_9)
```

```
Delay(); }
```

```
}
```