Ainsi, finalement nous n'avons pas grand chose à faire. Rendez-vous donc dans **arch/arm/mach-stm32/i2c.c** où nous pouvons constater qu'en fonction de la configuration du noyau, un certain nombre de choses sont d'ors et déjà en place :

```
#define I2C_STM32_REGS_SIZE     0x3FF

[...]

#if defined(CONFIG_STM32_I2C3)

#define I2C_STM32_DEV3_IRQ      72
#define I2C_STM32_DEV3_REGS     0x40005C00

static struct resource i2c_stm32_dev3_resources[] = {
  {
    .start  = I2C_STM32_DEV3_IRQ,
    .end    = I2C_STM32_DEV3_IRQ,
    .flags  = IORESOURCE_IRQ,
  },
  {
    .start  = I2C_STM32_DEV3_REGS,
    .end    = I2C_STM32_DEV3_REGS + I2C_STM32_REGS_SIZE,
    .flags  = IORESOURCE_MEM,
  },
};

static struct platform_device i2c_stm32_dev3 = {
  .name           = "i2c_stm32",
  .id             = 2,
  .num_resources  = ARRAY_SIZE(i2c_stm32_dev3_resources),
  .resource       = i2c_stm32_dev3_resources,
};

static struct i2c_stm32_data i2c_stm32_data_dev3 = {
  .i2c_clk        = 100000,
};

#endif  /* CONFIG_STM32_I2C3 */
```

On retrouve là toutes les informations nécessaires au pilote via le remplissage des structures dédiées. Il s'agit là d'utiliser l'API *platform device* qui est utilisée depuis longtemps dans le noyau Linux. En effet, les plateformes modernes utilisent principalement aujourd'hui des mécanismes rendant le matériel « découvrable » automatiquement (PCI, PCI-e, USB, etc) mais il existe toujours des périphériques qui doivent explicitement être déclarés pour que le noyau puisse s'en servir. C'est le cas de bien des fonctionnalités présentes sur les systèmes embarqués et en particulier du support i2c.

Pour déclarer un matériel, on rempli une structure **platform_device** qui sera utilisée ensuite avec **platform_device_register(struct platform_device *pdev)**. Cette structure renseigne le noyau sur le *plateforme device* à utiliser et ce, avec les informations fournies par une autre structure, **resource**, décrivant la manière d'accéder aux ressources afin que le pilote puisse utiliser effectivement le matériel. Ici nous trouvons donc dans **i2c_stm32_dev3_resources[]** le registre mappé en mémoire ainsi que l'interruption utilisée par le troisième bus i2c (**I2C3** dans le code, et plus tard **i2c-2**).

Ces déclarations faîtes, il ne nous reste plus qu'à enregistrer tout cela :

```
void __init stm32_i2c_init(void)
{
        int p = stm32_platform_get();

#if defined(CONFIG_STM32_I2C3)
  /*
   * Pass the device parameters to the driver
   */
  i2c_stm32_data_dev3.ref_clk = stm32_clock_get(CLOCK_PCLK1);
  platform_set_drvdata(&i2c_stm32_dev3, &i2c_stm32_data_dev3);

  /*
   * Register a platform device for this interface
   */
  platform_device_register(&i2c_stm32_dev3);
#endif

[...]

if (p == PLATFORM_STM32_STM32429_DISCO) {
    static struct i2c_board_info __initdata
            stm32f4_bdinfo_i2c3[] = {
            };
    i2c_register_board_info(2, stm32f4_bdinfo_i2c3,
                    sizeof(stm32f4_bdinfo_i2c3) /
                    sizeof (struct i2c_board_info));
  }
}
```

La fonction **stm32_i2c_init()** est appelée dans **stm32_platform.c** où sont centralisés toutes les initialisations de périphériques. En dehors de l'enregistrement du bus, un point intéressant concerne l'utilisation de **i2c_register_board_info**. Cette fonction permet d'enregistrer un périphérique sur le bus i2c. En effet, ce bus n'est pas énuméré matériellement au démarrage et il est nécessaire, si nous voulons qu'un pilote noyau prenne en charge un périphérique sur le bus, d'instancier le périphérique (voir fichier **Documentation/i2c/instantiating-devices**). Le présent code se voulant complet, nous avons inclus une liste vide.

En observant le code EmCraft, on se rend compte que pour leur SoM STM32F429, une EEPROM i2c Atmel AT24C512 est présente sur le bus. Il existe, par ailleurs, un pilote pour ce type d'EEPROM (**drivers/misc/eeprom/at24.c**) et pour que le composant soit alors pris en charge, le code suivant est utilisé :

```
#if defined(CONFIG_EEPROM_AT24)
  static struct i2c_board_info i2c_eeprom__stm32_som= {
       I2C_BOARD_INFO("24c512", 0x57)
  };
  i2c_register_board_info(0, &i2c_eeprom__stm32_som, 1);
#endif
```