LINUX SUR ARM CORTEX-M4 :

DÉVELOPPEMENT DE PILOTES I2C

par Denis Bodor

878D 735E A07A AD58 45DD 3F26 A1B2 2ECD 43C4 B3C8

La carte STM32F429IDISCOVERY dispose, comme nous l'avons vu, d'un écran LCD. Celui-ci est équipé d'un film tactile résistif connecté à un contrôleur STMPE811 interfacé en i2c. Voici l'occasion d'étudier comment uClinux gère ce type de bus de plus près, non sans un petit détour par les spécificités se rapportant aux GPIO.

Nous l'avons vu avec la gestion d'interruptions et la configuration de l'EXTI, le monde de l'ARM Cortex-M4 et du devkit STM32F429IDISCO est vaste et en mesure de satisfaire les développeurs et utilisateurs les plus assoiffés de connaissances techniques. Nous continuerons ici notre exploration en approchant le bus i2c de la plateforme. Mais avant cela, il nous est indispensable de parler de la gestion des GPIOs qui, comme vous pouvez vous en douter, est bien plus dense que celle de simples microcontrôleurs 8 bits comme les AVR d'Atmel, par exemple.

1 GPIO et iomux.c

Pour appréhender au mieux la gestion des GPIOs avec uCLinux, le plus simple est encore d'étudier le travail d'EmCraft dans le répertoire **arch/arm/mach-stm32/** avec, sous le coude, le *Reference manual* RM0090 de STMicroelectronics (1700 pages tout de même). Les GPIOs sont configurés après *reset* en tant qu'entrées flottantes. Le changement de configuration se fait par l'intermédiaire de registres mappés en mémoire. On retrouve l'implémentation correspondante dans **include/mach/iomux.h** :

```
/*
* GPIO registers base addresses
*/
#define STM32F2_GPIOA_BASE  (STM32_AHB1PERITH_BASE + 0x0000)
#define STM32F2_GPIOB_BASE  (STM32_AHB1PERITH_BASE + 0x0400)
#define STM32F2_GPIOC_BASE  (STM32_AHB1PERITH_BASE + 0x0800)
#define STM32F2_GPIOD_BASE  (STM32_AHB1PERITH_BASE + 0x0C00)
#define STM32F2_GPIOE_BASE  (STM32_AHB1PERITH_BASE + 0x1000)
#define STM32F2_GPIOF_BASE  (STM32_AHB1PERITH_BASE + 0x1400)
#define STM32F2_GPIOG_BASE  (STM32_AHB1PERITH_BASE + 0x1800)
#define STM32F2_GPIOH_BASE  (STM32_AHB1PERITH_BASE + 0x1C00)
#define STM32F2_GPIOI_BASE  (STM32_AHB1PERITH_BASE + 0x2000)

/*
* STM32F2/F4 GPIO control and status registers
*/
struct stm32f2_gpio_regs {
  u32  moder;   /* GPIO port mode                */
  u32  otyper;  /* GPIO port output type         */
  u32  ospeedr; /* GPIO port output speed        */
  u32  pupdr;   /* GPIO port pull-up/pull-down   */
  u32  idr;     /* GPIO port input data          */
  u32  odr;     /* GPIO port output data         */
  u16  bsrrl;   /* GPIO port bit set/reset low   */
  u16  bsrrh;   /* GPIO port bit set/reset high  */
  u32  lckr;    /* GPIO port configuration lock  */
  u32  afr[2];  /* GPIO alternate function       */
};
```

...avec une partie définie dans **include/mach/stm32.h** :

```
#define STM32_PERIPH_BASE     0x40000000
[...]
#define STM32_AHB1PERITH_BASE (STM32_PERIPH_BASE + 0x00020000)
```

La structure **stm32f2_gpio_regs** contient ainsi des entrées en **u32** permettant de définir le comportement des GPIOs :

- **moder** correspond au registre **GPIOx_MODER** permettant de configurer le sens d'une ligne (entrée, sortie, fonction alternative et analogique).
- **otyper** est **GPIOx_OTYPER** offrant le choix du type entre circuit *push-pull* (deux transistors) et en collecteur ouvert (*open drain*).
- **ospeedr** pour **GPIOx_OSPEEDR** et donc la vitesse de commutation entre *low* (2 Mhz), *medium* (10 Mhz) et *high* (50 Mhz).