LINUX SUR ARM CORTEX-M4 :

DÉVELOPPEMENT DE PILOTES POUR GPIO

Denis Bodor

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

Nous avons vu précédemment que l'installation d'uClinux sur un devkit STM32F429I-DISCO disposant de suffisamment de flash interne et d'une SDRAM de 64 Mbits ouvrait des perspectives très intéressantes en termes de développement. Le travail de quelques développeurs et de la société EmCraft permet ainsi de prendre en charge un certain nombre de périphériques. Il est maintenant temps de nous pencher sur l'intégration d'autres fonctionnalités...

Avant tout, il est de notre devoir de préciser que, bien que le travail d'EmCraft (https://github.com/EmcraftSystems/linux-emcraft) soit tout à fait remarquable concernant la prise en charge de plateformes Cortex-M3/M4, celui-ci n'est pas parfait. Entendez par là que le code, aussi fonctionnel soit-il, n'est pas du niveau de celui du noyau Linux officiel qualitativement parlant. En effet, on peut trouver un peu partout quelques raccourcis malheureux ou encore des traces évidentes d'importation de code d'autres projets et d'autres plateformes. Il est ainsi, par exemple, fait mention de SmartFusion (un produit Actel couplant FPGA et Cortex-M3) dans des commentaires du code `stm32_platform.c`, de Kinetis (une famille de MCU Freescale à coeur Cortex-M0 à M4) dans des noms de structures dans `mach-stm32/exti.c` prenant en charge le *External Interrupt/Event Controller* (EXTI) du STM32F4 ou encore du Blackfin d'Analog Devices dans le pilote de l'interface Ethernet STM32 (`blackfin_mii_bus` dans `drivers/net/arm/stm32_eth.c`)...

Un certain nombre de présupposés sont également fait en fonction de la plateforme utilisée, découlant directement des paramètres passés par le bootloader au noyau (retour de la fonction `stm32_platform_get()`). Le code, qu'il s'agisse de celui du dépôt GitHub d'EmCraft ou de celui de tmk/robutest (https://github.com/robutest/uclinux) est adapté à une utilisation pour une plateforme donnée, pour laquelle il a été initialement développé. Il ne semble pas conçu pour une utilisation générique ou une contribution amont dans uClinux. On remarquera d'ailleurs que la plupart des développements basés sur uClinux se font maintenant de la sorte, le projet officiel ne voyant plus beaucoup de mises à jour alors que le code est utilisé et adapté sur de nombreuses plateformes (ARM, MIPS, Blackfin, Microblaze, ColdFire, etc), souvent dans des versions plus récentes que celle proposée officiellement. Notez également que la majeure partie du code initial d'uClinux est maintenant présent et maintenue (par Greg Ungerer, le développeur ayant porté uClinux sur Coldfire) dans le noyau Linux standard (2.6). Le portage des modifications d'EmCraft et de tmk vers Linux 2.6.34.15 pourrait d'ailleurs être très intéressant en termes d'apprentissage, d'exploration du noyau et de contribution à ce dernier. Notez au passage que la version EmCraft disponible sur GitHub est également intégrée à leur « *STM32F429 Discovery Linux BSP* » vendu en ligne sur leur site au prix de $33 (nous laissons le soin au lecteur de déterminer la pertinence de cet achat au regard du code source disponible).

Quoi qu'il en soit, le noyau uClinux adapté au devkit STM32F429 est donc déjà passablement lié avec une ou plusieurs plateformes. Il est donc acceptable d'ajouter notre touche personnelle dans ces mêmes conditions en intégrant alors simplement quelques fonctions et définitions supplémentaires dans le code existant. Ceci nous permettra, sur la base existante, de découvrir très simplement les différents périphériques de la plateforme et, par la même occasion, d'affirmer la polyvalence et la légitimité d'utiliser uClinux sur la carte malgré ces faibles ressources.