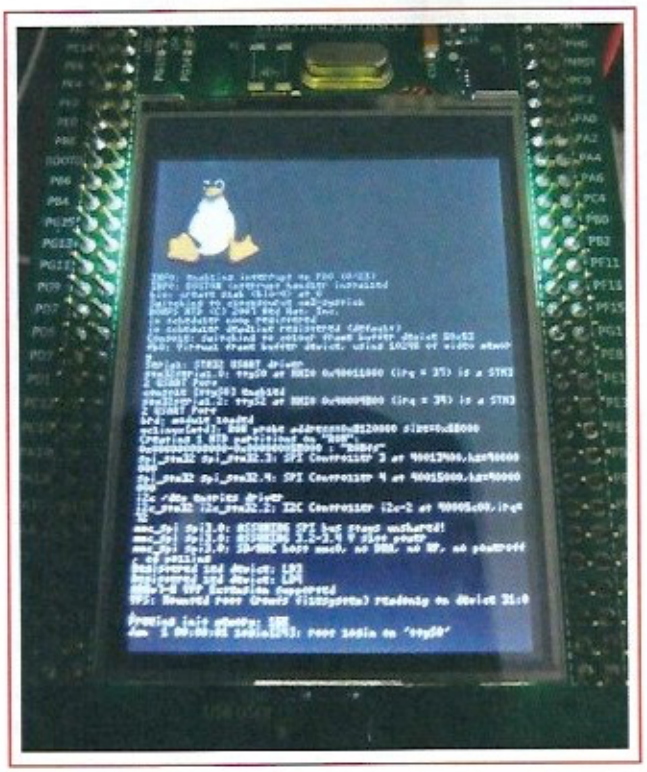
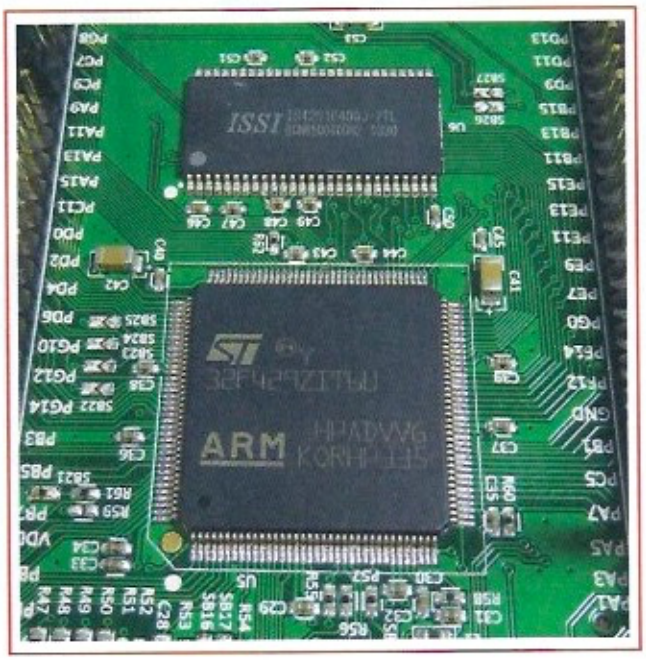
applications une fois l'ensemble de votre temps et votre énergie investi dans le projet, ceci ne vous conduira nul part. C'est également un élément qu'il faudra expliquer à votre client s'il tient absolument à utiliser un STM32F407 (avec SRAM externe via l'interface FSMC), par exemple, pour une utilisation avec µClinux.

2 µClinux sur STM32F429-DISCO

En novembre dernier, un utilisateur postait un message discret sur l'un des forum ST. Ce développeur turc se faisant appeler Robutest/tmk y annonce sa satisfaction de voir que la société EmCraft maintient une version d'µClinux plus plateforme STM32 et que, sur cette base, il s'est demandé s'il pouvait adapter ce portage au nouveau devkit STM32F429IDISCO. Le message fait référence à une vidéo Youtube postée par ses soins présentant la carte démarrant le système avec une console en framebuffer sur l'écran LCD inclus au devkit.



Le travail est impressionnant puisque cela inclus le portage du bootloader U-Boot, du noyau et de Busybox, mais surtout la prise en charge du support pour l'interface LCD du microcontrôleur. Dans la discussion qui s'en suit sur le forum, cette démontration découle sur une mise en ligne des sources, quelques questions/réponses et surtout une synthèse des travaux sous la forme d'un autre dépôt GitHub créé par un autre membre du forum (jserv) : https://github.com/jserv/stm32f429-linux-builder. Ce dernier, au vu des difficultés de certains à reproduire la démonstration et à compiler les différents éléments, a développer ce qu'on pourrait appeler un « gros Makefile » (ou un nano-buildroot) pour automatiser l'ensemble du processus. C'est précisément ce système que nous allons utiliser dans un premier temps afin d'obtenir un système viable rapidement que nous pourrons ensuite personnaliser.



2.1 Chaîne de compilation

La première étape indispensable consiste à disposer d'un cross-compilateur adapté. Force est de constater que si l'on cherche à utiliser un tel compilateur qui semble facile à construire à partir des sources, on s'engage dans un processus de recherche long et débouchant généralement sur des informations très anciennes et donc un compilateur tout aussi âgé. Même des outils comme Crosstool-NG semblent avoir oublié de prendre en compte µClinux. La solution la plus simple est alors de se diriger vers un cross-compilateur qu'on téléchargera sous forme binaire pour être utilisé localement dans une installation minimal et sans interférence avec le système de gestion de paquets de sa distribution. Celui généralement conseillé et majoritairement utilisé provient de CodeSourcery, maintenant Mentor Graphics, désigné sous l'appellation commerciale *Mentor Graphics Sourcery Tools*. Ces outils sont destinés à être utilisés avec l'environnement de développement *Sourcery CodeBench*. Bien entendu, ici, nous n'avons que faire d'un environnement complet et ne souhaitons disposer que d'une chaîne de compilation pour un hôte GNU/Linux