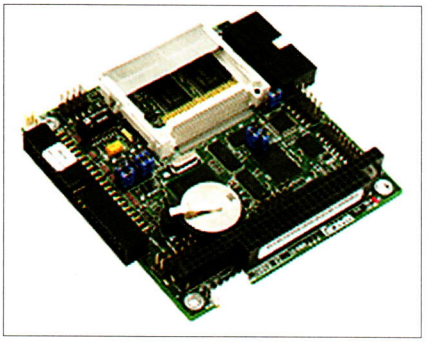

Figure 6. Carte VIPER de ARCOM

mémoire spéciale. Il peut s'agire soit d'une mémoire NOR de 2Mo, soit d'une partition d'une mémoire NAND si le processeur permet d'exécuter du code issu d'une telle mémoire dès le *boot* (ce qui est rarement le cas des petites architectures).

Oui, mais ou s'exécute le code du noyau et des applications ? En RAM bien entendu, sauf qu'il s'agit bien souvent de SDRAM, bien moins chère comparée aux technologies modernes. La mémoire SDRAM a ceci de particulier qu'elle doit être rafraîchie en permanence par le processeur faute de quoi ses données seront immanquablement perdues.

Le démarrage d'un système uClinux est donc le suivant :

- Le BIOS, le noyau et le romfs sont logés en mémoire NOR (les 2 premiers Mo de l'espace adressable),
- Le BIOS est exécuté au démarrage et initialise les différents périphériques. Il lance ensuite l'exécutable auto extractible du noyau uClinux. Celui-ci se décompresse en SDRAM et s'exécute automatiquement,
- Lors du démarrage du noyau, uClinux identifie le romfs qui se trouve en *flash NOR*, ainsi que les différents périphériques (*Ethernet*, MMC, NAND, USB, I2C, ...) et termine par la création du processus *init* et l'exécution du script */etc/rc*. Il peut ainsi monter des extensions du système de fichiers en flash NAND (périphérique MTD).

Comme nous l'avons soulignés précédemment, ce type d'architecture ne dispose pas d'un bus PCI, donc les périphériques doivent être gérés d'une façon différente. En fait le processeur possède dans ce cas un « wrapper mémoire », capable d'activer des signaux de sélection de périphériques en fonction de l'adresse choisie. Ainsi toutes les accès aux adresses 0x06000000 et au delà dans une limite de 32Mo par exemple seront destinés au contrôleur Ethernet.

Génération du noyau et des applications

Dans l'embarqué il est plus que jamais indispensable de reconstruire son noyau pour l'utilisation spécifique qui en sera faite. Cette étape peut effrayer le novice, mais des configurations types existent pour vous aider à reconstruire votre noyau et reflasher votre cible.

Tout comme pour votre PC, vous pourrez configurer le noyau de votre cible à l'aide de la commande `make xconfig` et sélectionner ainsi les périphériques que vous souhaitez utiliser.

Dans le cas d'un noyau Linux 2.6, vous aurez par la suite à lancer la commande `make`. A contrario le noyau 2.4 nécessitera de lancer une commande `make dep` au préalable.

Il vous faudra ensuite compiler un shell et l'ensemble des utilitaires de base comme *mount* ou *ifconfig*, sans parler du processus *init* sans lequel vous ne pourrez exécuter aucun processus !

Le mieux est encore d'utiliser la Busybox, véritable couteau suisse informatique qui vous permettra d'obtenir la plupart des binaires nécessaires à votre application et ceci en quelques clicks de souris. En effet la sélection des programmes se fait de façon graphique à l'aide d'une commande `make menuconfig` et la compilation pour une architecture croisée ne présente aucune difficulté.

La distribution uClinux a toutefois une particularité : elle comporte non seulement le noyau mais aussi l'ensemble des applications embarquées adaptées à uClinux, comme par exemple la Busybox, les outils de gestion réseau, le serveur graphique Nano-X ou encore le serveur *http Boa*. Enfin, un simple *make* suffit à générer à la fois le noyau, les différentes librairies (comme la uClibc), les applications et le romfs.

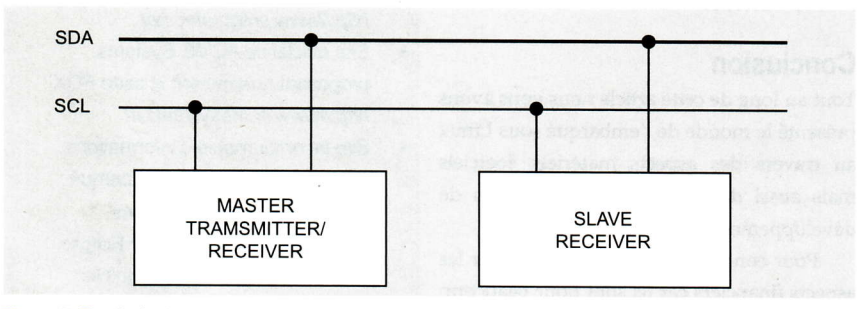

Figure 7. Topologie du bus I2C™

Cet article explique

Cet article est une introduction au monde de l'embarqué, et surtout de la présence et de l'utilisation du systeme Linux dans ce milieu. Il expose les grandes catégories de matériels utilisés dans l'embarqué ainsi que les différents types de distributions associées, ainsi que les outils mis librement a disposition.

Débuguer son application

Bon nombre de développeurs utilisent encore comme seul et unique outil une simple console et *vi* ou *vim*. Il est vrai qu'avec d'aussi simples outils il est possible de réaliser 100% du travail à effectuer, c'est-à-dire l'édition, la compilation et le transfert sur la cible.

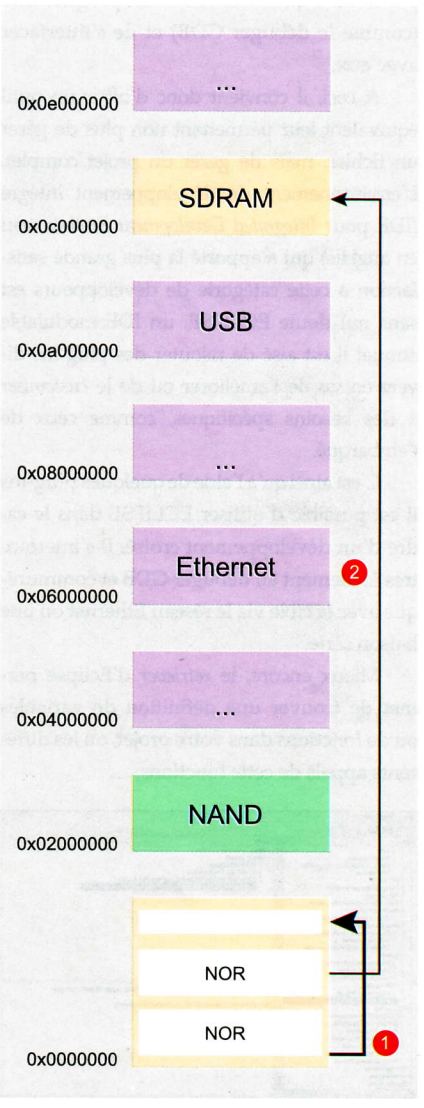

Figure 8. Exemple d'organisation mémoire d'un système embarqué