Notre plate-forme est maintenant complète ou presque : elle possède le hardware qui va permettre de s'interfacer au réseau CAN tout en offrant des interfaces utiles pour le reste de nos développements (USB, Ethernet, RS232, ...), elle utilise maintenant un driver *can3680* modifié qui permettra de piloter de façon performante le SJA1000 et enfin elle dispose d'un noyau uClinux qui lance automatiquement le programme de notre choix.

Il ne nous reste donc plus qu'à réaliser ce programme. Nous le réaliserons à l'aide d'un simple éditeur et nous le compilerons avec un compilateur croisé ARM (`arm-elf-gcc`). Pour le transférer sur la cible et faire quelques essais, rien de tels que des connections FTP et TELNET. Nous nous assurerons donc de la présence des 2 services ainsi que de l'inet sur notre cible afin de pouvoir développer et tester nos programmes de façon plus conviviales.

Réalisation de l'application

Nous allons en fait réaliser un petit CANalyseur, appareil destinée habituellement à espionner un bus CAN afin de permettre de suivre l'évolution de certaines variables du réseau ou encore de les stocker dans des fichiers de log.

Pour cela nous n'allons pas utiliser de filtres hardwares (dans le SJA1000) ou software (au niveau du driver) afin d'être sûr de recevoir dans notre application toutes les trames CAN qui véhiculent sur le bus.

Nous allons donc créer un programme dont le cœur sera essentiellement le contenu du Listing 1. Si vous êtes connecté à un réseau un peu chargé, vous risquez de recevoir certaines trames à une période de 10ms, ce qui vous fera plusieurs centaines de trames affichées par seconde ... et ce n'est évidemment pas l'effet recherché. Dans ce cas, modifiez le programme pour n'afficher que les trames qui changent.

Disons en somme que lors de vos premiers essais vous chercherez à identifier les ID des trames de votre réseau. Ensuite, vous pourrez toujours vous focaliser sur des trames une à une.

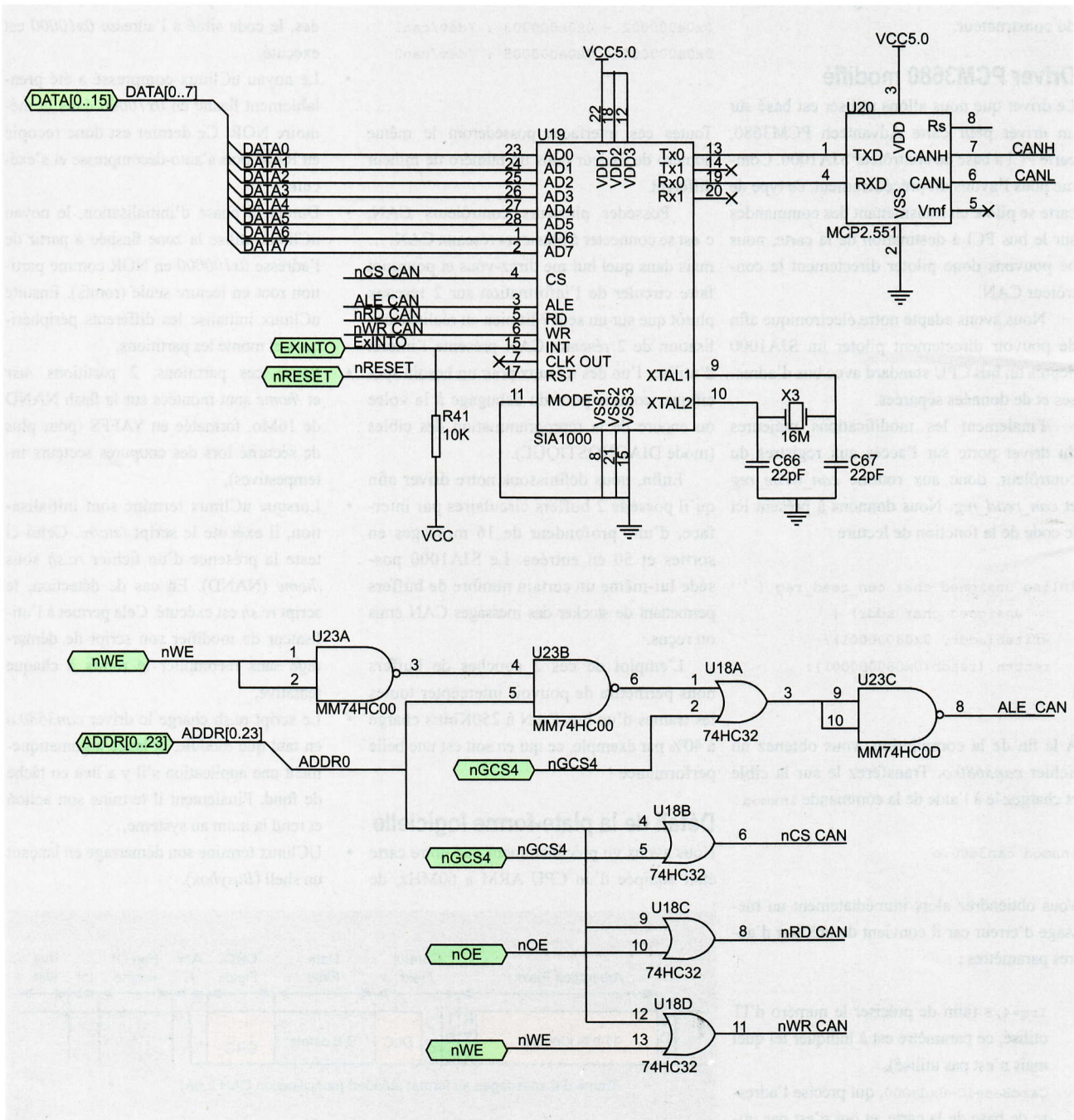

Figure 4. Adaptation électronique du SJA1000