QUEL ACTE STUPIDE ?

Ceci vous est totalement invisible lorsque vous utilisez une carte Arduino, mais le microcontrôleur qui s'y trouve possède une configuration. Celle-ci prend la forme de « fusibles » (*fuses*) qui sont en réalité une série de bits stockés dans trois registres particuliers (le terme « fusible » n'est pas adapté, car l'écriture n'est pas irréversible. C'est un héritage du passé où les bits étaient effectivement définitivement inscrits). Ces bits permettent par exemple de régler l'utilisation d'un bootloader appelé « séquence d'initialisation » dans l'IDE Arduino (programme de démarrage, lancé juste avant votre croquis en mémoire flash), la vitesse de démarrage ou encore, la source d'horloge du composant.

Cette dernière configuration est souvent celle qui peut conduire aux problèmes. Le microcontrôleur AVR d'une carte Arduino utilise un oscillateur à quartz à 16 Mhz pour fonctionner et il est donc configuré dans ce sens. Mais un AVR peut également fonctionner avec d'autres sources comme un résonateur céramique, un signal externe ou encore des quartz de fréquence différentes (32,768 Khz par exemple). Il peut également ne pas utiliser du tout de composant extérieur et reposer entièrement sur une horloge interne, moins précise, mais permettant d'avoir un circuit plus simple.

Une fois le microcontrôleur configuré pour une source, celui-ci fonctionne totalement sur cette base, et ceci vaut pour le mode « programmation » du composant. En clair, si vous configurez votre microcontrôleur pour une source externe à quartz de haute fréquence alors que ce composant n'est pas présent, la configuration sera effectivement inscrite, mais les tentatives de programmation ultérieures ne fonctionneront pas. Du moins pas temps que le quartz en question (et ces deux condensateurs) n'est pas installé. Vous ne pourrez donc plus revenir en arrière, car le microcontrôleur ne vous répondra simplement plus.

La configuration des fusibles d'un microcontrôleur AVR peut donc déboucher sur une situation bloquante, et la source d'horloge n'est pas le seul paramètre qui peut conduire à ce type de chose. En l'absence de circuit correspondant à la configuration, il ne reste plus qu'une seule solution : le HVSP pour *High Voltage Serial Programming*. Une technique de programmation permettant de mettre le microcontrôleur dans un mode particulier pour reprendre la main et reconfigurer les fusibles correctement.

La plupart des programmeurs existants ne disposent pas de ce mode de programmation particulier nécessitant, entre autres, l'application d'une tension de 12 volts sur la broche « reset ». Il faut alors soit passer par un montage maison, soit utiliser un programmeur comme l'Atmel AVR Dragon.

Conclusion : si vous commencez à jouer avec les fusibles des microcontrôleurs, mieux vaut avoir un programmeur qui supporte le HVSP.

Dans cette réalisation, nous allons économiser un ATmega328p en le remplaçant par un Atmega168 moins coûteux et surtout traînant inutilement depuis des années dans un de mes tiroirs. L'idée ici n'est pas réellement d'économiser au maximum, mais de simplement éviter d'utiliser une carte Arduino pour cet usage unique. Si vous recherchez une manière la plus économique possible de programmer un microcontrôleur Atmel AVR, il existe des dizaines de produits « programmeur USB ISP » compatibles STK500 ou USBasp pour quelques euros. À l'opposé, si le prix vous importe peu ou que vous souhaitez le programmeur le plus polyvalent et le plus puissant possible, vous avez le Atmel-ICE de chez Atmel capable de programmer énormément de composants Atmel (MegaAVR, TinyAVR, Xmega, SAM, etc.), mais il vous en coûtera environ 60 euros. À ce prix, vous pourrez vous sortir de presque n'importe quelle situation découlant d'un acte stupide. Le programmeur Atmel AVR Dragon est également un très bon choix pour cela. Si vous êtes entre les deux solutions, cet article est fait pour vous.

Terminons cette introduction en précisant que les opérations qui vont suivre reposent sur les fonctionnalités offertes par la nouvelle version 1.6.4 de l'environnement Arduino et en particulier l'ajout de cartes décrit par ailleurs dans ce numéro. Nous utiliserons, en effet, le support « Barebones ATmega Chips » de C. Rodrigues, puis le support « attiny » de D. Mellis pour le premier essai.