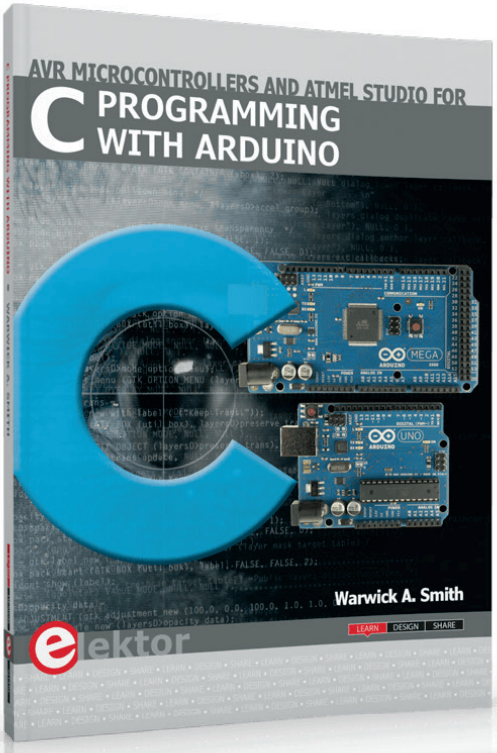
mises au point & mises à jour

corrections – questions – réponses

Clemens Valens (Elektor Labs)

Mises à jour et compléments d'informations sur des articles publiés par Elektor, avec des tuyaux, des astuces, des conseils ingénieux et des réponses à des questions d'intérêt général posées par des lecteurs.

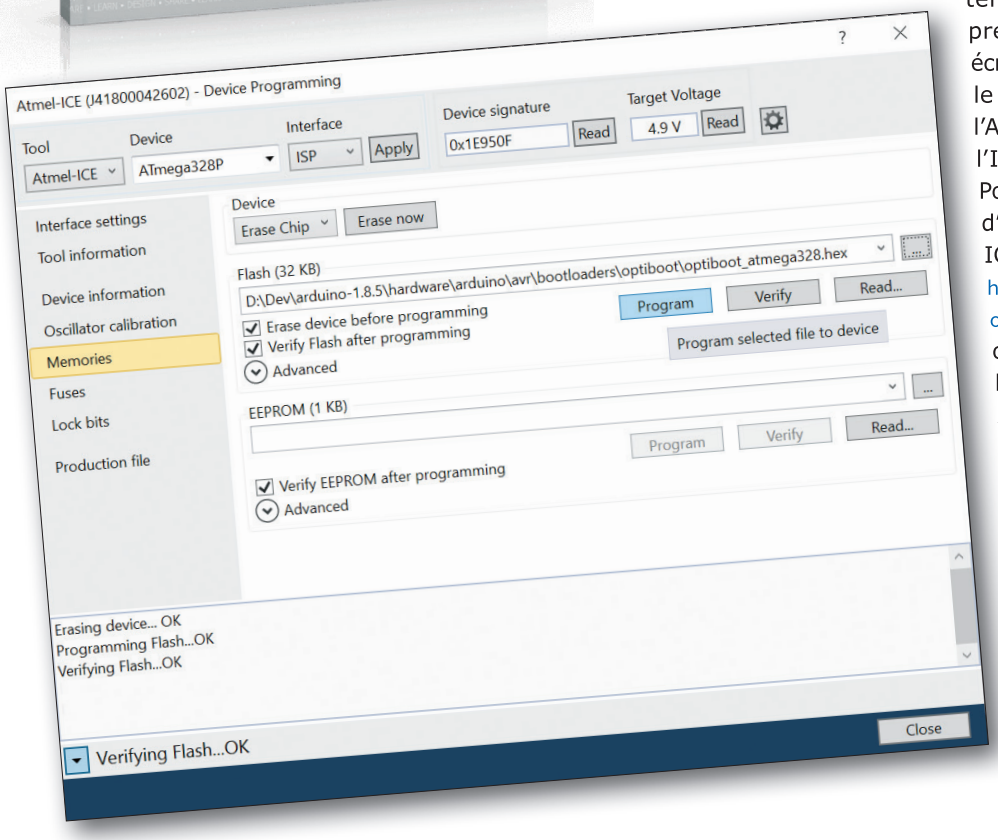


Arduino Uno : quand l'amorçage capote

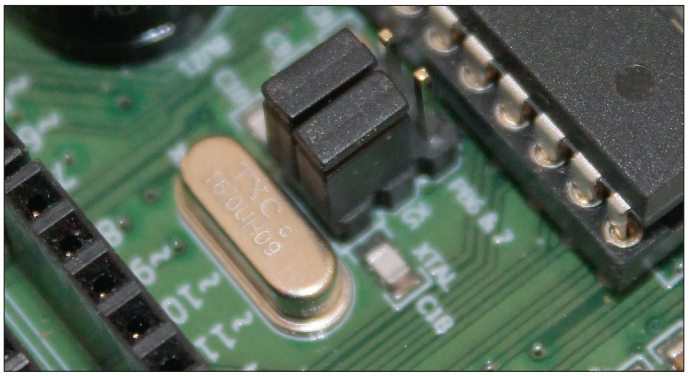
Q : J'ai acheté le livre *C Programming with Arduino* d'Elektor et m'y suis mis avec enthousiasme. J'ai aussi fait les frais d'un programmateur Atmel ICE de Microchip qui fonctionne bien avec Atmel Studio 7.0 (AS7), l'environnement (IDE) utilisé dans le livre. J'ai constaté que l'Arduino Uno ne peut plus être programmé par l'IDE Arduino. J'ai essayé un autre Uno qui lui fonctionne toujours avec l'IDE Arduino. Reconnecter cette carte, redémarrer l'ordinateur ou le logiciel ni aucune autre manipulation similaire n'a donné de résultat. J'ai cherché la solution en vain sur plusieurs forums.

R : C'est le chargeur d'amorce (dit *bootloader*) Arduino qui transforme un microcontrôleur (µC) quelconque en µC Arduino. Ce petit logiciel réside à demeure dans la mémoire de l'unité centrale et il est capable de charger dans cette même mémoire un programme d'application, reçu la plupart du temps via une connexion sérielle. Et cela sans s'écraser lui-même ! L'IDE Arduino connaît évidemment ce *bootloader* et utilise le port série (via USB) pour programmer la carte Arduino.

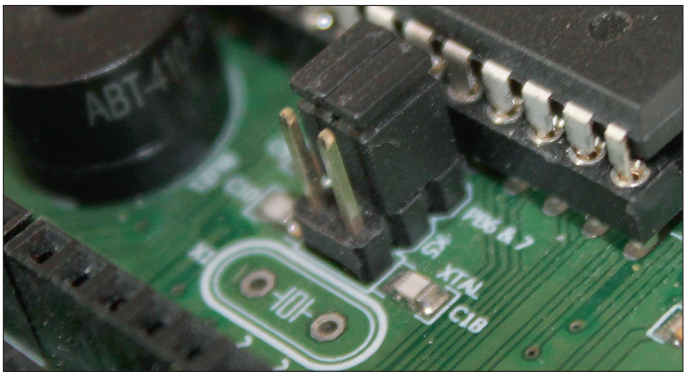
En revanche, Atmel Studio ne connaît ni Arduino ni son chargeur d'amorce. C'est pourquoi il faut, pour inscrire un programme dans la mémoire du µC, un adaptateur de programmation comme l'ICE Atmel. Celui-ci se connecte à l'interface de programmation interne du µC (ISP) pour accéder directement à sa mémoire. En l'absence de précautions adéquates, cette opération écrase le contenu de la mémoire, y compris le chargeur d'amorçage. Voilà pourquoi l'Arduino Uno a cessé de fonctionner avec l'IDE Arduino : son *bootloader* a disparu !



Pour reprogrammer le chargeur d'amorce d'Arduino, on peut utiliser AS7 avec Atmel ICE. Le fichier nécessaire est `[Arduino]\hardware\arduino\avr\bootloaders\optiboot\optiboot_atmega328.hex` où `[Arduino]` est le dossier contenant le fichier `arduino.exe`. Une fois dans AS7, ouvrez *'Device Programming'* (Maj-Ctrl-P). Puis, dans l'onglet *Memories* de la section *Flash,* cliquez sur le bouton *Browse for file* et naviguez jusqu'au fichier HEX du *bootloader*. Cliquez sur *Program* pour faire écrire le logiciel d'amorçage dans le µC. Il y a plusieurs façons de configurer AS7 pour Arduino afin d'éviter de supprimer le logiciel d'amorçage chaque fois que vous cliquez sur *Program*. Ces méthodes vont de l'installation



La carte Playground AVR avec un quartz de 16 MHz.



AVR Playground running from its internal 8-MHz RC-oscillator.

d'extensions pour AS7 à l'adaptation des paramètres du *linker*. Vous trouverez sur l'internet la méthode qui vous conviendra le mieux.

Q : Est-il possible d'utiliser l'ATmega328 d'une carte Arduino Uno avec la carte Playground AVR et *vice versa* ?

R : L'Arduino Uno et la carte Playground AVR utilisent le même µC ATmega328, mais vous ne pouvez pas simplement les interchanger. Cela est dû à leur oscillateur d'horloge. Celui de l'Arduino Uno a une horloge de 16 MHz asservie par son quartz de 16 MHz. Sur la carte Playground AVR, c'est l'oscillateur RC interne de 8 MHz du µC qui bat la mesure.

Vous pouvez remplacer le µC d'une carte Uno par un µC de Playground AVR. L'Uno fonctionnera alors comme un Playground AVR fonctionnant à 8 MHz. Cependant, l'inverse, c'est-à-dire remplacer le µC d'une Playground par celui d'une Uno, n'est possible que lorsque l'AVR Playground est équipé d'un quartz de 16 MHz (X1) et que les cavaliers de K5 sont sur les broches 1-3 et 2-4. Une fois ces conditions remplies, le Playground AVR fonctionnera comme un Uno.

La programmation du chargeur d'amorce dans le µC d'une carte compatible Arduino tient compte de la fréquence d'horloge du µC afin de configurer la vitesse du port série utilisé pour le téléchargement des croquis : donc, si la fréquence d'horloge change, la vitesse du port série change aussi. L'IDE Arduino ne permet malheureusement pas à l'utilisateur de spécifier lui-même le débit du port série pour le téléchargement des croquis. Lorsque vous sélectionnez une carte Arduino Uno dans l'IDE Arduino, l'IDE suppose qu'elle fonctionne à 16 MHz et nécessite un débit de téléchargement de 115 200 bauds. Avec un Uno qui n'est pas cadencé à 16 MHz, le téléchargement d'un croquis échouera donc. De même, un Playground AVR doit, pour l'IDE, fonctionner à 8 MHz avec un débit de téléchargement de 57 600 bauds.

La simplicité de l'IDE Arduino est appréciée par la plupart de ses utilisateurs, mais elle implique des restrictions. L'utilisateur aventureux tentera de modifier le fichier `boards.txt` où sont réunies toutes les informations sur les cartes. Sachez qu'il peut y avoir plus d'un de ces fichiers dans votre installation de l'EDI Arduino. Les clés à modifier sont `[Board].upload.speed` et `[Board].build.f_cpu` (où `[Board]` est le nom de la carte). L'IDE doit être relancé pour prendre en compte les modifications apportées à ce fichier.

www.elektormagazine.com/labs/avr-playground-129009-2

nouveau GPS pour une nouvelle horloge Nixie précise

Le module GPS *Maestro* A2035-H utilisé dans ce projet n'est plus produit. Cependant, *Lantronix*, qui a acquis *Maestro* (été 2019), propose le A2235-H, plus récent, qui, bien que plus petit, s'adapte à notre circuit imprimé. Il suffit d'ajouter deux résistances de polarisation de 2,2 kΩ entre ses broches I²C et sa broche 1,8 V. Pour éviter tout court-circuit entre le blindage du module et un via en dessous de celui-ci, intercalez une isolation (Kapton).

www.elektormagazine.com/labs/150189-6-digit-nixie-clock

190379-B-02