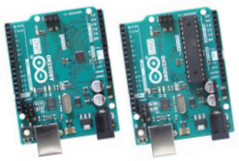
ULTIMATE Arduino Uno Hardware Manual

Extrait : chargeur d'amorçage du microcontrôleur principal

Warwick A. Smith (Afrique du Sud)

Cet article contient environ un quart de chapitre du livre intitulé *Ultimate Arduino Uno Hardware Manual* de Warwick A. Smith, publié par Elektor en juin 2021. Au fil du temps, l'Arduino Uno est devenu un « produit de base ». Comme il est bon marché, l'utilisateur est peu regardant sur les caractéristiques matérielles et il peut se retrouver en difficulté pour faire fonctionner son application. Le chargeur d'amorçage de l'Uno et les réglages des fusibles sont obscurs, la plupart des questions techniques reçues chez Elektor portent sur ces sujets. Les réponses « définitives » données par ce nouveau livre d'Elektor seront donc appréciées.

Note de l'éditeur : cet article est un extrait du livre *Ultimate Arduino Uno Hardware Manual* formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine *Elektor*. Puisque cet article est extrait d'une publication plus vaste, certains termes peuvent faire référence à des passages du livre d'origine situés ailleurs. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré « Des questions, des commentaires ? ».



Chargeur de démarrage du microcontrôleur principal

Sur une carte *Arduino Uno*, si le µcontrôleur principal (*ATmega328P*) est remplacé par une puce vierge, il faudra y implanter le chargeur. C'est faisable avec *Microchip Studio* et un programmateur USB externe connecté au connecteur *ICSP*. La programmation a lieu de la même manière que pour l'*ATmega16U2*, mais il faut utiliser le connecteur ICSP à l'extrémité de la carte, côté opposé au connecteur USB. De cette façon, il est aussi possible de sauvegarder le chargeur déjà implanté dans un ATmega328P.
Le détail de ces actions figure dans les sections qui suivent et incluent les réglages des fusibles pour l'ATmega328P utilisé sur une carte Arduino Uno. Le chargeur peut aussi être restauré rapidement à partir de l'EDI Arduino, mais avec Microchip Studio installé, cela ne fonctionne pas à cause d'un conflit de pilotes.

Sauvegarde du µlogiciel de l'ATmega328P avec Microchip Studio

Par précaution, le µlogiciel de l'ATmega328P peut être lu et copié sur la carte Arduino Uno et sauvegardé dans des fichiers HEX et BIN avec l'utilitaire de programmation des puces de Microchip Studio. C'est très pertinent si on utilise une carte clone qui peut avoir un chargeur de démarrage différent de celui des cartes Arduino Uno authentiques.
Branchez un programmateur USB adéquat à l'ordinateur, puis le câble avec le connecteur femelle à 6 broches du programmateur dans le connecteur ICSP à l'extrémité de la carte Arduino Uno. Consultez le brochage de l'embase ICSP pour ATmega328P qui indique où se trouve la broche 1. Alimentez l'Arduino Uno soit par l'USB, soit avec une alimentation externe.
Démarrez Microchip Studio et ouvrez l'utilitaire *Device Programming* en cliquant sur l'icône *Device Programming* dans la barre d'outils du haut, ou en sélectionnant *Tools Device Programming* sur le menu d'accueil de Microchip Studio. Dans la boîte *Device Programming*, sélectionnez le programmateur USB correct sous *Tools*, l'*ATmega328P* sous *Device*, vérifiez qu'*ISP* (*In-System Programmer*) est sélectionné sous *Interface*, et cliquez sur le bouton *Apply*. Pour vérifier que tout est bien configuré et que Microchip Studio peut se connecter à la cible, cliquez sur le bouton *Read* sous *Device Signature*. La signature de la cible et la consigne de tension seront lues et affichées dans la boîte de dialogue *Device Programming*.
Dans la colonne de gauche de la boîte *Device Programming*, cliquez sur *Memories*. Dans la section *Flash* de *Memories* de la boîte de dialogue, cliquez sur *Read...* pour naviguer jusqu'à un dossier où stocker le fichier HEX récupéré de la mémoire Flash de l'ATmega328P. Au bas de la boîte *Save as*, entrez un nom valide au choix pour le fichier HEX. Cliquez sur *Save* quand c'est terminé.
L'utilitaire *Device Programming* lit la mémoire Flash de l'ATmega328P et la sauvegarde dans un fichier HEX du nom choisi. Ce fichier HEX peut être utilisé pour restaurer le µlogiciel original de l'ATmega328P à tout moment. Le fichier HEX comprend toute la mémoire Flash et donc le croquis utilisateur programmé dans la mémoire Flash au moment de la création du fichier HEX.

Micrologiciel du chargeur de démarrage Optiboot

Les cartes Arduino Uno utilisent désormais le chargeur **Optiboot** dans le µcontrôleur ATmega328P. Il remplace un ancien chargeur. Optiboot occupe un segment de 512 octets de mémoire Flash, l'ancien chargeur avait besoin de 2 Ko. Il y a donc 1,5 Ko de mémoire Flash de plus disponibles pour les croquis de l'utilisateur.
Le code source C et les fichiers HEX d'Optiboot se trouvent dans le dossier Arduino EDI, sous le chemin :

```
arduino-1.8.13\hardware\arduino\avr\bootloaders\
    optiboot
```

où le dossier racine aura un nom différent pour les différentes versions de l'EDI Arduino. Les mêmes fichiers figurent sur les pages GitHub Arduino [1].
Le projet Optiboot est sur GitHub à l'adresse *github.com/Optiboot/optiboot* et pour en savoir plus, le wiki Optiboot est à l'adresse *github.com/Optiboot/optiboot/wiki*.
Optiboot fait clignoter la LED L de l'Arduino au démarrage. Il commence alors à exécuter le croquis présent, ou attend qu'un nouveau croquis soit chargé si l'EDI de l'Arduino le réinitialise. Pour plus de détails, voir le Wiki Optiboot [2].

Restauration du chargeur de démarrage

Microchip Studio et un programmateur USB externe permettent de restaurer/charger le chargeur d'amorçage Optiboot sur l'ATmega328P d'origine ou un vierge, comme cela a déjà été décrit dans ce chapitre pour l'ATmega16U2. Pour se connecter à l'ATmega328P depuis Microchip Studio, utilisez la même méthode que pour sauvegarder le chargeur d'amorçage de la section « Backing up the ATmega328P Firmware with Microchip Studio ».
Pour charger Optiboot sur l'ATmega328P de l'Arduino Uno, connectez le programmateur USB au connecteur ICSP à l'extrémité de la carte comme décrit ci-dessus. Ouvrez l'utilitaire *Device Programming* dans Microchip Studio, et de même, connectez le programmateur et l'ATmega328P. Cliquez sur l'élément *Memories* dans la colonne de gauche de la boîte *Device Programming*. Cliquez sur *[...]* dans la section *Flash* de la boîte de dialogue pour en ouvrir une autre qui sert à naviguer vers le fichier HEX à charger dans l'ATmega328P. Le fichier HEX de restauration du chargeur de démarrage peut être soit celui créé par la sauvegarde du chargeur, comme décrit à la section 5.4.1, soit le fichier HEX correct provenant de :

```
arduino-1.8.13\hardware\arduino\avr\bootloaders\
    optiboot
```

qui pour l'Arduino Uno est *optiboot_atmega328.hex*. Sélectionnez le bon fichier dans la boîte de dialogue, puis cliquez sur *Open*. De retour dans la boîte *Device Programming*, cliquez sur *Program* dans la section *Flash* de celle-ci. Si tout est bien configuré, la mémoire Flash sera programmée avec le fichier HEX sélectionné.
Si le chargeur de démarrage a été installé sur un ATmega328P vierge, les fusibles de celui-ci doivent être réglés correctement avant de

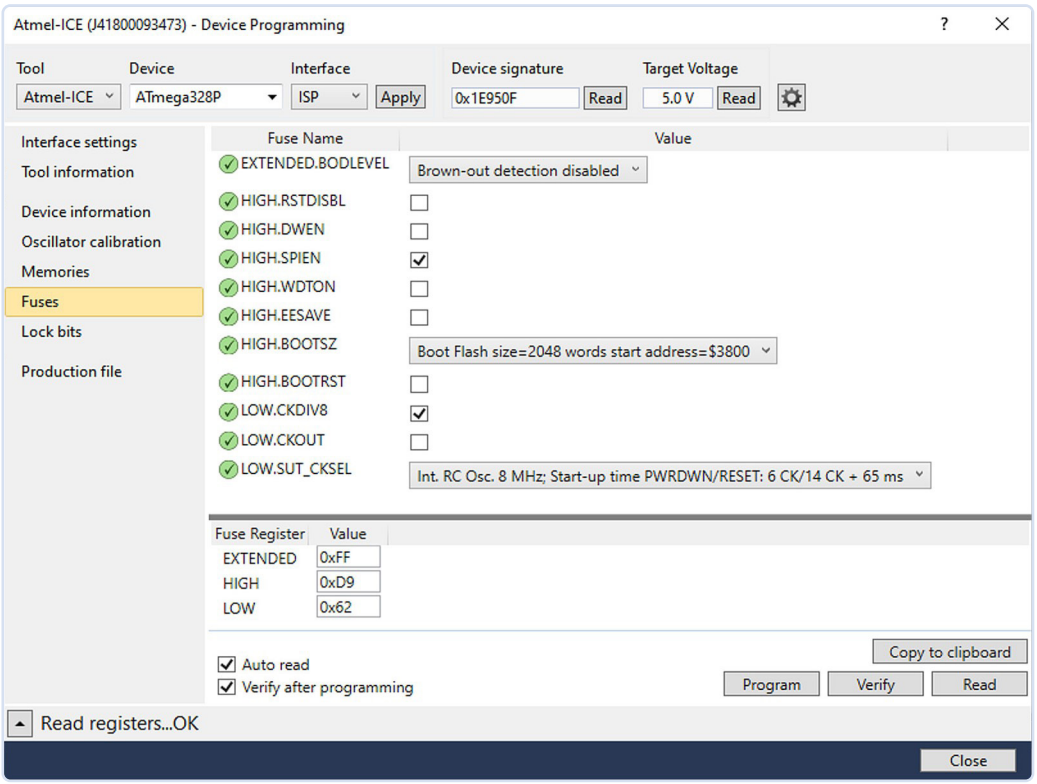


Figure 1. Paramètres par défaut des fusibles pour un ATmega328P vierge.

continuer, comme décrit plus loin. Si les fusibles sont déjà réglés correctement, fermez la boîte *Device Programming* de Microchip Studio et débranchez l'alimentation de l'Arduino Uno. Déconnectez le programmateur USB de l'Arduino, puis remettez l'Arduino sous tension. Si les fusibles sont déjà bien réglés, testez le chargeur de démarrage tout juste implanté en ouvrant l'EDI Arduino et en chargeant un croquis de test avec la méthode normale du câble USB.

Réglage des fusibles de l'ATmega328P

Avec un ATmega328P vierge, il faut d'abord charger le *bootloader* Optiboot comme décrit ci-dessus. Les fusibles de l'ATmega328P doivent ensuite être réglés pour être utilisés dans l'Arduino Uno. Les valeurs des fusibles peuvent être lues et réglées avec l'utilitaire *Device Programming* de Microchip Studio. Pour lire et régler les fusibles, d'abord branchez un programmateur USB externe sur le connecteur ICSP à l'extrémité de l'Arduino Uno comme décrit plus haut. Dans Microchip Studio, ouvrez l'utilitaire *Device Programming*, choisissez les paramètres corrects pour l'outil, la puce et l'interface comme déjà décrit, puis cliquez sur *Apply* pour vous connecter. Dans la colonne de gauche de la boîte *Device Programming*, cliquez sur *Fuses*.
La **figure 1** montre ce à quoi il faut s'attendre pour les valeurs des fusibles d'un µcontrôleur ATmega328P vierge dans la boîte *Device Programming*. Les valeurs par défaut des registres de fusibles sont :

EXTENDED	0xFF
HIGH	0xD9
LOW	0x62

Ces fusibles doivent être programmés selon les valeurs de la **figure 2** et du **tableau 1**. Les valeurs des fusibles suivants doivent être modifiées :

Fusible	Remplacer par :
EXTENDED.BODLEVEL	Brown-out detection at VCC=2.7V
HIGH.EESAVE	(optional, see next page for an explanation)
HIGH.BOOTSZ	Boot Flash size=256 words; start address=$3F00
HIGH.BOOTRST	checked
LOW.CKDIV8	unchecked
LOW.SUT_CKSEL	Ext. Crystal Osc. 8.0- MHz; Start up time PWRDWN/ RESET: 16K/14 CK + 65ms

Une fois ces réglages effectués, les valeurs des registres de fusibles du bas de la boîte de dialogue sont :

EXTENDED	0xFD
HIGH	0xDE (EESAVE unchecked) or 0xD6 (EESAVE checked)
LOW	0xFF

En général, EESAVE est coché sur les anciennes cartes Arduino Uno R3, mais décoché sur les nouvelles. Si la case EESAVE (fusible HIGH.EESAVE) est cochée dans la boîte de dialogue, elle protège le contenu de l'EEPROM si la puce est effacée.
Après avoir effectué les changements, cliquez sur *Program* près du bas de la boîte de dialogue. Les nouvelles valeurs des fusibles seront alors appliquées à l'ATmega328P. Pour continuer, cliquez sur *OK* dans la boîte d'avertissement. Fermez la boîte *Device Programming*,

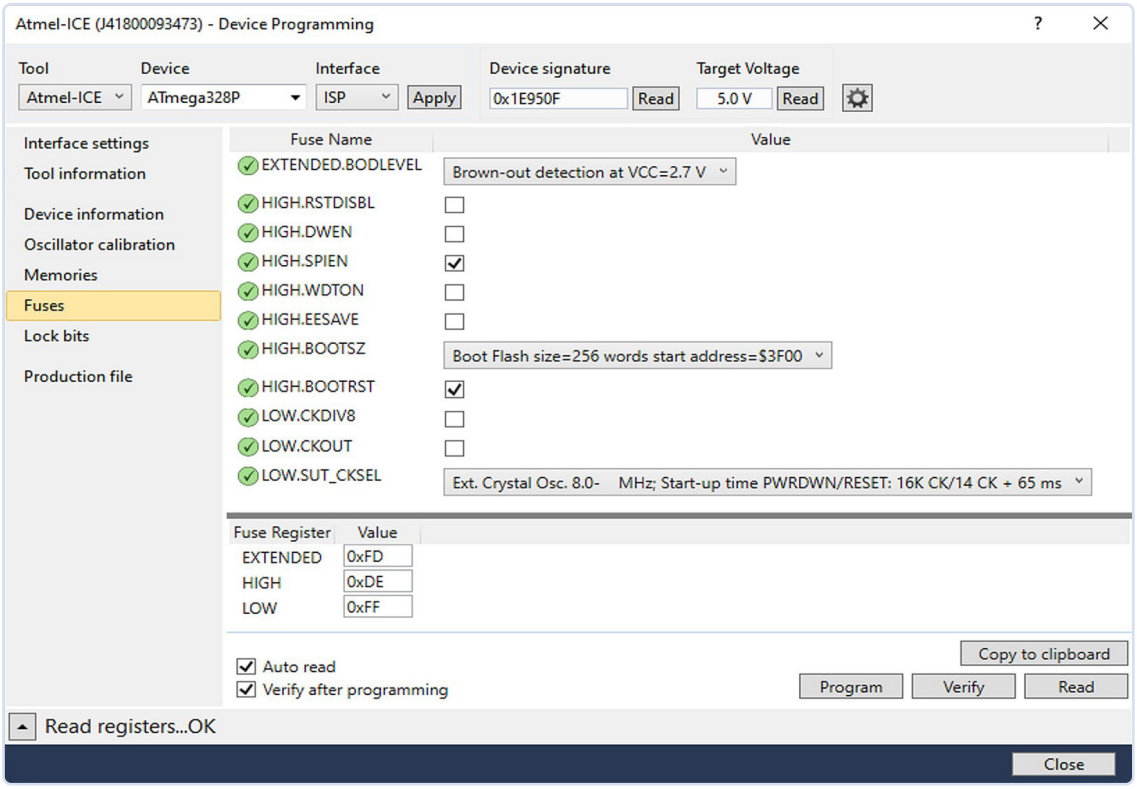


Figure 2. Réglage des fusibles de l'ATmega328P pour l'Arduino Uno.

Tableau 1. Réglage des fusibles de l'Arduino Uno ATmega328P.

Nom du fusible	Valeur
EXTENDED.BODLEVEL	Brown-out detection level at VCC=2.7 V
HIGH.RSTDISBL	unchecked
HIGH.DWEN	unchecked
HIGH.SPIEN	☑
HIGH.WDTON	unchecked
HIGH.EESAVE	unchecked on new boards
HIGH.BOOTSZ	Boot Flash size=256 words, start address=$3F00
HIGH.BOOTRST	☑
LOW.CKDIV8	unchecked
LOW.CKOUT	unchecked
LOW.SUT_CKSEL	Ext.Crystal Osc. 8.0 MHz; Start-up PWRDWN/RESET: 16K CK/14 CK + 65 ms

débranchez l'alimentation de l'Arduino Uno, puis le programmateur USB. Branchez l'Arduino Uno au port USB du PC, puis chargez-y un croquis à partir de l'EDI Arduino pour vérifier que le chargeur fonctionne et que les paramètres des fusibles sont corrects.

Le pont de soudure RESET-EN

Certains des programmateurs AVR USB ont un débogueur, notamment l'AVR Dragon (Microchip ne le vend plus) et l'Atmel-ICE (il remplace les anciens programmateurs USB). Notez que l'AVRISP mkII (Microchip ne le vend plus) n'a pas de capacité de débogage, c'est un programmateur au sens strict. C'est aussi le cas des programmateurs proposés aux amateurs, tels que l'USBasp et l'USBtinyISP, qui sont des appareils limités à la programmation. Le débogage fait référence à la capacité du programmateur/débogueur USB à fonctionner conjointement avec un logiciel tel que Microchip Studio, et permettre à l'utilisateur, d'accéder au µcontrôleur principal de l'Arduino pour visualiser le contenu de sa mémoire et de ses registres internes, ainsi que de parcourir le code source en une seule étape et d'y insérer des points d'arrêt. La version 1.8.13 de l'EDI Arduino, la version en cours au moment de la rédaction, ne dispose pas de cette capacité.

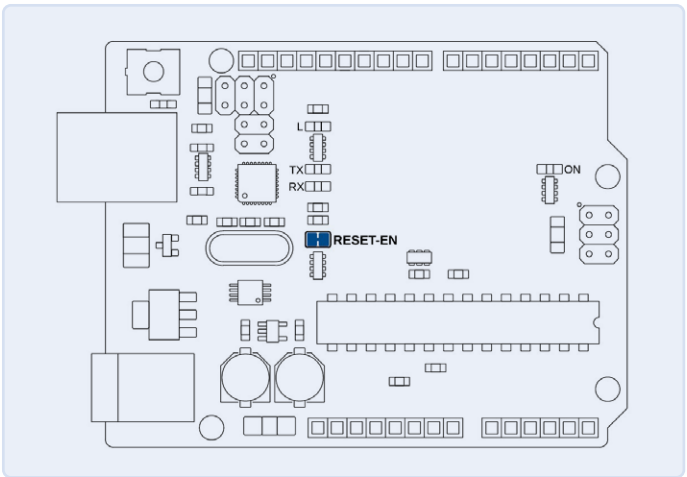
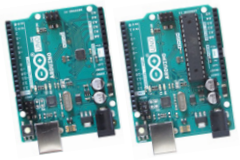


Figure 3. Emplacement du pont de soudure RESET-EN sur l'Arduino Uno.



Pour exploiter les capacités de débogage du programmateur/débogueur USB, on le branche au connecteur ICSP comme on le ferait normalement pour programmer la mémoire Flash du dispositif AVR cible, ou pour lire les valeurs des fusibles. Un programme en C ou C++ peut ensuite être chargé sur l'AVR à l'aide de Microchip Studio. Microchip Studio peut alors être utilisé pour déboguer l'AVR cible.

Le système de débogage du µcontrôleur ATmega328P s'appelle **debugWIRE** et permet au logiciel de débogage de se connecter au µcontrôleur cible à travers le connecteur ICSP au moyen d'un programmateur USB capable de débogage. Pour que debugWIRE fonctionne sur l'ATmega328P des cartes Arduino Uno, il faut déconnecter le circuit de réinitialisation du signal RESET. Les cartes Arduino Uno ont un pont appelé RESET-EN formé de deux pastilles soudées reliées par une piste de circuit imprimé. Il faut couper cette piste afin de déconnecter une partie du circuit de réinitialisation du signal RESET lors de l'utilisation de debugWIRE. Pour rétablir la connexion, ressoudez les deux pastilles ensemble La **figure 3** montre où sont les pastilles de soudure RESET-EN sur l'Arduino Uno. Répétons que cette fonction ne sera utilisée que par les utilisateurs avancés qui programment la carte en C avec Microchip Studio. Les lecteurs désireux d'apprendre le langage C et programmer les cartes Arduino seront intéressés par le livre *C Programming with Arduino* (ISBN 978-1-907920-46-2) du même auteur, aux éditions Elektor International Media. Ce livre comprend un chapitre sur le débogage des cartes Arduino Uno et Mega 2560 à l'aide d'Atmel Studio (depuis remplacé par Microchip Studio).

Autres méthodes de programmation du micrologiciel

Des méthodes alternatives pour programmer le chargeur d'amorçage d'un Arduino Uno sont présentées sur le site *playground Arduino*, section Bootloader [3]. On y trouve des infos sur l'utilisation d'un second Arduino comme programmateur d'un Arduino cible ainsi que sur l'utilisation d'autres programmateurs.

L'exploitation d'un Arduino comme ISP est aussi décrite sur le site [4]. Le chapitre 2, section 2.8, du livre contient plus d'informations sur le chargement des croquis sur l'Arduino Uno avec un programmateur externe et l'EDI Arduino. Consultez la même section pour des infos sur la restauration du chargeur d'amorçage sur le µcontrôleur principal avec un programmateur externe et l'EDI Arduino.

Enfin, voir [5] pour des informations supplémentaires sur la programmation *DFU* (*Device Firmware Update*) de l'ATmega16U2.

(210345-04)

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (warwsmi@axxess.co.za) ou contactez Elektor (redaction@elektor.fr).

Contributeurs
Texte et illustrations : **Warwick A. Smith**
Rédaction : **Jan Buiting**
Traduction : **Yves Georges**
Mise en page : **Sylvia Sopamena**

LIENS

[1] Optiboot sur Github : www.github.com/arduino/ArduinoCore-avr/tree/master/bootloaders/optiboot
[2] Wiki Optiboot : www.github.com/Optiboot/optiboot/wiki/HowOptibootWorks
[3] Chargeur de démarrage Arduino : https://playground.Arduino.cc/Main/ArduinoCoreHardware/#Bootloader
[4] Arduino comme ISP : https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoISP
[5] Programmation DFU (*Device Firmware Update*) de l'ATmega16U2 : https://www.arduino.cc/en/Hacking/DFUProgramming8U2