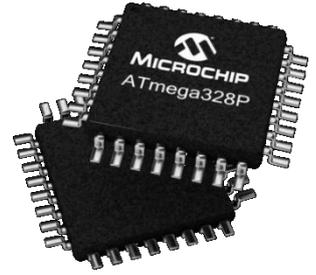




Démarrage d'un ATmega328 et Boot Arduino



1. Objectif

Comment démarrer et entrer en communication avec un ATmega328 que vous venez d'installer sur votre carte et le préparer à communiquer avec Arduino?

2. Conditions de démarrage

En dehors de défauts de conception de la carte, le démarrage d'un ATmega ne requiert que peu de conditions :

- Une alimentation de la bonne valeur sur les bonnes broches,
- Un reset au bon niveau,
- Un oscillateur en état de fonctionner,
- Des bits de configuration (*fuse bits*) correctement programmés.

Les trois premières conditions sont déterminées par le matériel et donc le schéma de la carte.

Les bits de configuration peuvent être configurés si le matériel le permet.

3. Caractéristiques électriques

3.1. *Alimentation*

La documentation nous montre dans un premier temps de faibles contraintes tant qu'on ne dépasse pas 6V.

29.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to $V_{CC}+0.5V$
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins	200.0mA



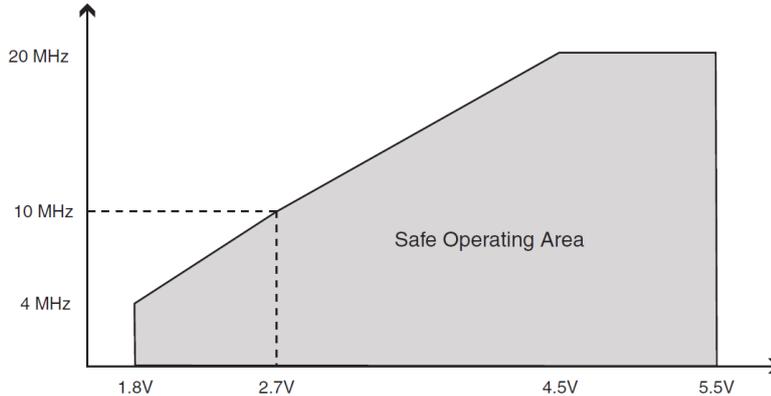
3.2. Tension d'alim et fréquence d'horloge

L'ATMega peut travailler sous basse tension d'alim à la condition d'être cadencé à une fréquence compatible avec cette alim :

Speed Grades

Maximum frequency is dependent on V_{CC} . As shown in Figure 29-1, the Maximum Frequency vs. V_{CC} curve linear between $1.8V < V_{CC} < 2.7V$ and between $2.7V < V_{CC} < 4.5V$.

Figure 29-1. Maximum Frequency vs. V_{CC}



Comme on le voit, une alim supérieure à 4.5V permet d'atteindre la fréquence max autorisée pour ce circuit.

Par contre une alimentation inférieure à 4.5V nécessite un cadencement à une fréquence inférieure à 20MHz.

Les cartes Arduino alimentées en 5V sont cadencées à 16MHz.

Les cartes Arduino alimentées en 3.3V sont cadencées à 8MHz.

Inutile donc d'espérer faire fonctionner à 20MHz un ATMega alimenté en 3.3V.

3.3. Le reset

La broche RESET permet de mettre le μC en conditions de démarrage et bloque son fonctionnement lorsqu'elle est maintenue à une tension proche de 0V.

Pour autoriser le fonctionnement du μC il faut donc que la tension sur cette broche (après mise sous tension) soit supérieure à une certaine valeur dépendant de l'alim.

Pour faire vite : elle doit être supérieure à $0.9 \times V_{CC}$ en temps normal.

Table 29-11. Reset, Brown-out and Internal Voltage Characteristics⁽¹⁾

Symbol	Parameter	Min.	Typ	Max	Units	
V_{POT}	Power-on Reset Threshold Voltage (rising)	1.1	1.4	1.6	V	
	Power-on Reset Threshold Voltage (falling) ⁽²⁾	0.6	1.3	1.6	V	
SR_{ON}	Power-on Slope Rate	0.01		10	V/ms	
V_{RST}	\overline{RESET} Pin Threshold Voltage	$0.2 V_{CC}$		$0.9 V_{CC}$	V	
t_{RST}	Minimum pulse width on \overline{RESET} Pin			2.5	μs	
V_{HYST}	Brown-out Detector Hysteresis		50		mV	
t_{BOD}	Min. Pulse Width on Brown-out Reset		2		μs	
V_{BG}	Bandgap reference voltage	$V_{CC}=2.7$ $T_A=25^\circ C$	1.0	1.1	1.2	V
t_{BG}	Bandgap reference start-up time	$V_{CC}=2.7$ $T_A=25^\circ C$	40	70	μs	
I_{BG}	Bandgap reference current consumption	$V_{CC}=2.7$ $T_A=25^\circ C$	10		μA	

Notes: 1. Values are guidelines only.
 2. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)



3.4. Les Fuse Bits

Ils permettent de configurer l'ATMega et ses conditions de démarrage.
 Leur positionnement est dépendant de l'application.
 Pour Arduino leur positionnement est fixé selon le type de carte.
 Il y a 3 octets de *Fuse Bits* : *Low*, *High* et *Extended*.

Table 28-6. Extended Fuse Byte for ATmega328/328P

Extended Fuse Byte	Bit No	Description	Default Value
–	7	–	1
–	6	–	1
–	5	–	1
–	4	–	1
–	3	–	1
BODLEVEL2 ⁽¹⁾	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1 ⁽¹⁾	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0 ⁽¹⁾	0	Brown-out Detector trigger level	1 (unprogrammed)

Note: 1. See [Table 29-12 on page 305](#) for BODLEVEL Fuse decoding.

Table 28-8. Fuse High Byte for ATmega328/328P

High Fuse Byte	Bit No	Description	Default Value
RSTDISBL ⁽¹⁾	7	External Reset Disable	1 (unprogrammed)
DWEN	6	debugWIRE Enable	1 (unprogrammed)
SPIEN ⁽²⁾	5	Enable Serial Program and Data Downloading	0 (programmed, SPI programming enabled)
WDTON ⁽³⁾	4	Watchdog Timer Always On	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed), EEPROM not reserved
BOOTSZ1	2	Select Boot Size (see Table 27-7 on page 275 , Table 27-10 on page 276 and Table 27-13 on page 277 for details)	0 (programmed) ⁽⁴⁾
BOOTSZ0	1	Select Boot Size (see Table 27-7 on page 275 , Table 27-10 on page 276 and Table 27-13 on page 277 for details)	0 (programmed) ⁽⁴⁾
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

- Notes:
1. See ["Alternate Functions of Port C" on page 85](#) for description of RSTDISBL Fuse.
 2. The SPIEN Fuse is not accessible in serial programming mode.
 3. See ["WDTCSR – Watchdog Timer Control Register" on page 54](#) for details.
 4. The default value of BOOTSZ[1:0] results in maximum Boot Size. See ["Pin Name Mapping" on page 286](#).



Table 28-9. Fuse Low Byte

Low Fuse Byte	Bit No	Description	Default Value
CKDIV8 ⁽⁴⁾	7	Divide clock by 8	0 (programmed)
CKOUT ⁽³⁾	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed) ⁽¹⁾
SUT0	4	Select start-up time	0 (programmed) ⁽¹⁾

Table 28-9. Fuse Low Byte (Continued)

Low Fuse Byte	Bit No	Description	Default Value
CKSEL3	3	Select Clock source	0 (programmed) ⁽²⁾
CKSEL2	2	Select Clock source	0 (programmed) ⁽²⁾
CKSEL1	1	Select Clock source	1 (unprogrammed) ⁽²⁾
CKSEL0	0	Select Clock source	0 (programmed) ⁽²⁾

- Note:
1. The default value of SUT1...0 results in maximum start-up time for the default clock source. See [Table 9-12 on page 34](#) for details.
 2. The default setting of CKSEL3...0 results in internal RC Oscillator @ 8MHz. See [Table 9-11 on page 34](#) for details.
 3. The CKOUT Fuse allows the system clock to be output on PORTB0. See ["Clock Output Buffer" on page 36](#) for details.
 4. See ["System Clock Prescaler" on page 36](#) for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Ce qui est critique au moment du démarrage c'est le choix de la source d'horloge (CKSEL).

En effet, en sortie d'usine, l'ATMega est configuré sur son oscillateur interne (note 2).

Cela lui permet de fonctionner sans oscillateur externe ou malgré un oscillateur externe défectueux. Si, lors de la programmation du *Fuse Low Byte*, on choisit une source d'horloge externe sans que celle-ci soit présente ou fonctionnelle, on ne peut plus communiquer avec l'ATMega bien que cela ait été possible la première fois sur l'oscillateur interne. Il faut alors fournir une horloge externe valide pour reprendre la main sur le circuit.

On ne détaillera pas ici les autres *Fuse Bits*.



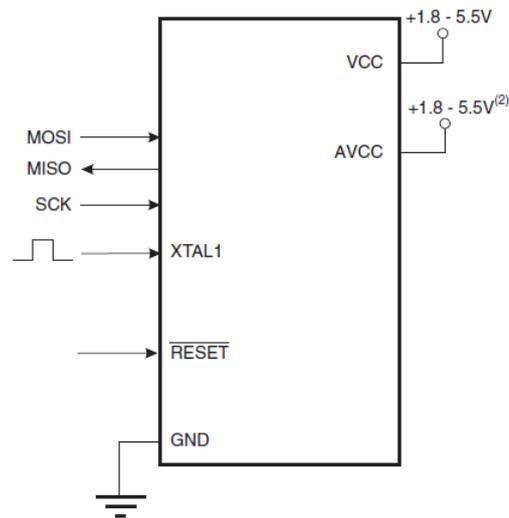
4. Programmation en ICSP

On peut programmer l'ATMega lorsqu'il est soudé sur une carte à condition d'avoir prévu les connections pour utiliser le mode ICSP (In Circuit System Programming) et de disposer d'un programmeur comme l'AVRISP.

Un connecteur à 6 broches est généralement prévu pour cela.



Figure 28-7. Serial Programming and Verify⁽¹⁾



Comme on le voit ce mode nécessite :

- Une alimentation
- Une source d'horloge
- La prise en main des signaux MOSI, MISO, CLK et Reset.

5. Procédure de démarrage

5.1. *Inspection visuelle, continuité*

Avant la mise sous tension, une inspection visuelle est nécessaire pour détecter les court-circuits ou coupures flagrants.

Un test de continuité sur les bornes d'alim peut permettre de détecter une alim en court-circuit.

En cas de doute le schéma et l'ohmmètre sont les outils indispensables.

5.2. *Test des alims*

Si possible, seul l'ATMega est câblé :

- Brancher une alim de labo entre VCC et GND réglée sur 0V et limitée à 80mA (ou une valeur compatible avec la consommation des autres composants sur la carte).
- Augmenter la tension progressivement jusqu'à Vcc en vérifiant que la consommation ne dépasse pas les valeurs prévues (selon configuration de la carte).
- Si consommation excessive : chercher les court-jus, composants inversés...

La consommation nulle :

- Si la consommation est nulle l'ATMega n'est peut-être pas alimenté : vérifier VCC et GND.



5.3. Test des signaux principaux

- Contrôler la tension sur les broches vitales :

Signal	VCC	GND	RST
Pin	1, 6, 18	3, 21	29
Tension	Vcc	0V	Vcc

5.4. Communication avec le programmeur AVRISP

Si l'ATMega est alimenté et reçoit un niveau correct sur RESET il devrait pouvoir être pris en main au travers d'un programmeur.

Nous utiliserons pour cela le logiciel ATMEL Studio (v7.0 à l'heure actuelle). C'est un outil de développement qui sert à beaucoup de choses mais nous utiliserons seulement sa capacité à piloter l'AVRISP.

Une fois le programmeur connecté en USB et le logiciel lancé on accède à la programmation de circuits par l'icone  ou par `\tools\Device programming` ou par `CTRL+SHIFT+P`.

On choisit AVRISP MK2 dans `Tools` et le microcontrôleur dans `Device` (ATMega328P pour une Arduino Uno).

5.5. Connexion à la carte

Pour prendre en main l'ATMega il faut que l'AVRISP soit connecté au connecteur de programmation de la carte (attention au brochage) et que la carte soit alimentée.

Si tout va bien, une led verte s'allume sur l'AVRISP.

Sinon une led orange ou rouge s'allume.

5.6. Test de la communication

Un clic sur `Read` (à coté de `Device signature`) et la signature du μ C doit apparaitre.

5.7. Paramètres et Bootloader Arduino

Vous trouverez les paramètres à configurer pour votre type de carte Arduino dans le fichier :

```
boards.txt
```

situé dans :

```
<répertoire_d'installation_arduino>\hardware\arduino\avr.
```



Voici l'extrait concernant l'Arduino Uno :

```
uno.name=Arduino/Genuino Uno

uno.vid.0=0x2341
uno.pid.0=0x0043
uno.vid.1=0x2341
uno.pid.1=0x0001
uno.vid.2=0x2A03
uno.pid.2=0x0043
uno.vid.3=0x2341
uno.pid.3=0x0243

uno.upload.tool=avrdude
uno.upload.protocol=arduino
uno.upload.maximum_size=32256
uno.upload.maximum_data_size=2048
uno.upload.speed=115200

uno.bootloader.tool=avrdude
uno.bootloader.low_fuses=0xFF
uno.bootloader.high_fuses=0xDE
uno.bootloader.extended_fuses=0xFD
uno.bootloader.unlock_bits=0x3F
uno.bootloader.lock_bits=0x0F
uno.bootloader.file=optiboot/optiboot_atmega328.hex

uno.build.mcu=atmega328p
uno.build.f_cpu=16000000L
uno.build.board=AVR_UNO
uno.build.core=arduino
uno.build.variant=standard
```

On y trouve :

- **Les Fuse Bits :**

```
uno.bootloader.low_fuses=0xFF
uno.bootloader.high_fuses=0xDE
uno.bootloader.extended_fuses=0xFD
```

- **Les Lock Bits :**

```
uno.bootloader.unlock_bits=0x3F
uno.bootloader.lock_bits=0x0F
```

- **Le type de μ C :**

```
uno.build.mcu=atmega328p
```

- **La fréquence d'horloge :**

```
uno.build.f_cpu=16000000L
```

Etc.



5.8. Programmation des Fuse/Lock Bits

Un clic sur Fuses devrait permettre de lire l'état des Fuse Bits et d'obtenir quelque chose qui ressemble à ça :

Simulator - Device Programming

Tool: Simulator | Device: ATmega328P | Device signature: 0x1E950F

Fuse Name	Value
EXTENDED.BODLEVEL	Brown-out detection disabled
HIGH.RSTDISBL	<input type="checkbox"/>
HIGH.DWEN	<input type="checkbox"/>
HIGH.SPIEN	<input checked="" type="checkbox"/>

Fuse Register	Value
EXTENDED	0xFF
HIGH	0xD9
LOW	0x62

Starting operation read registers
Reading register EXTENDED...OK
Reading register HIGH...OK
Reading register LOW...OK
Read registers...OK

Read registers...OK

Il nous faut donc programmer les Fuse Bits comme demandé dans le fichier `boards.txt` soit :

```
uno.bootloader.low_fuses=0xFF  
uno.bootloader.high_fuses=0xDE  
uno.bootloader.extended_fuses=0xFD
```

Un clic sur Program : le bits devraient être positionnés et un `Read registers...OK` confirmer le succès de l'opération.

Si la communication avec l'ATMega devient impossible après la programmation des Fuse Bits c'est peut être que l'oscillateur externe (paramétré dans Fuses Low) ne fonctionne pas. coup d'oscillo sur XTAL1 et XTAL2 permettra de lever le doute (broches 7 et 8 sur un ATMega328 en TQFP32).



5.9. Lock Bits

Les lock bits servent à protéger la flash d'un effacement accidentel lors du fonctionnement. En Arduino cela revient à protéger le Bootloader.

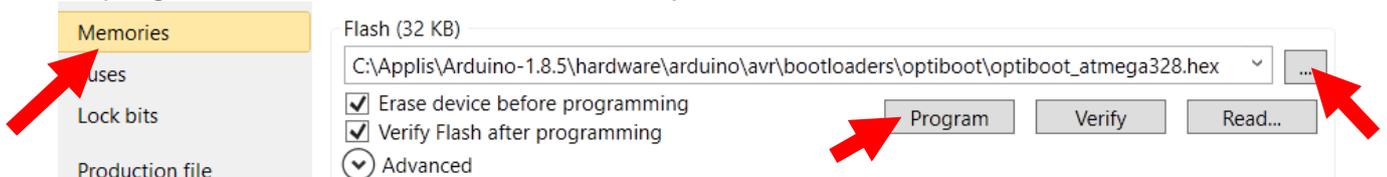
Il ne faut donc pas les positionner avant d'avoir programmé le Bootloader.

Une erreur dans le positionnement des Lock Bits empêcherait une programmation future du μC .

Nous ne les programmerons pas. Cela ne nuira pas au fonctionnement de notre carte.

5.10. Programmation du Bootloader

La programmation du Bootloader se fera à partir de `Memories` :



6. Test de programmation Arduino

Le `ATMega` a été configuré et le Bootloader installé. Vérifions que l'on peut téléverser du code depuis Arduino.

Pour cela il faut :

- Un IDE Arduino installé et fonctionnel,
 - Un adaptateur USB/Série (style FTDI) et ses drivers fonctionnels,
 - L'adaptateur connecté à l'`ATMega` : au moins GND, TX, RX, DTR et VCC si la carte n'est pas alimentée ailleurs,
 - L'`ATMega` (la carte) doit être alimenté.
-
- Chargez un petit programme simple dans Arduino : typiquement `analogReadSerial`.
 - Téléversez vers votre carte.

Si DTR est câblé et que votre carte le prend en charge le programme doit se téléverser.

Si DTR n'est pas câblé ou non fonctionnel on peut toujours déclencher le téléversement par un appui sur Reset au moment du téléversement.

Les leds Tx et Rx de l'adaptateur clignotent durant le téléversement.

Pour `analogReadSerial`, on doit voir les valeurs défilier dans le moniteur série.

6.1. Ca téléverse pas

6.1.1. Adaptateur série

- Adaptateur déconnecté de la carte et connecté à Arduino (drivers ok et bon port série),
- Ouvrez le moniteur série.
- Emettez n'importe quoi.
- La led Tx de l'adaptateur doit clignoter.

6.1.2. Alim depuis l'adaptateur série

- L'adaptateur série fonctionne seul.



- Branchez-le à votre carte.
- Si la led d'alim de l'adaptateur s'éteint : vérifiez les alims de votre carte ou alors la consommation de votre carte est trop élevée pour l'adaptateur seul.

6.1.3. Liaison Tx USB (TXO) vers Rx μ C

Emettez depuis le moniteur série : si la led Tx ne clignote plus : vérifiez la ligne Tx (Rx du μ C) depuis le connecteur jusqu'au μ C. Faites de même pour la ligne Rx.

Branchez un oscillo et observez le Rx du μ C sur les pattes du μ C.

Vous devez voir les trames série (0-Vcc).

6.1.4. Liaison Tx μ C vers Rx USB (RXI)

Lors d'une tentative de téléversement, la led TX de l'USB clignote et on voit les trames arriver à la patte Rx du μ C.

Observez en même temps la patte Tx du μ C.

Si vous observez une tentative d'émission du μ C (des trames très atténuées) : cherchez un problème sur la ligne Tx μ C jusqu'au connecteur de communication.

6.2. *Ca ne marche toujours pas*

Il va falloir faire de l'électronique.