STEMKRAF – ATMEGA328 fuse bits
https://github.com/teaksoon/stemkraf

The “fuse” bits contains settings (0 or 1) that are used by the chip to determine how it should function.

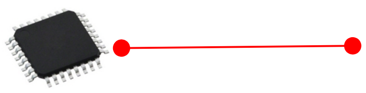
There are total of 19 fuse bits in the Atmega328 micro-controller chip. They can be accessed via three(3) seperated bytes, they are commonly known as

1. low fuse bits (1 byte – 8 bits)
2. high fuse bits (1 byte – 8 bits)
3. extended fuse bits (1 byte – 8 bits, only 3 bits are used here)

Depending on are needs, the fuse can be set by the factory or by us.

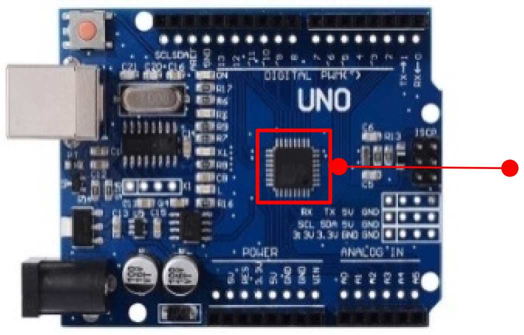
However, be very careful when you change the fuse settings because you may render the chip “unusable”. Most of us does not have the tools to set back the fuse to a “useable” settings again

Here, we are just reading and displaying the content of the fuse with our program. No harm will be done.



Atmega328 micro-controller stand alone chip (QFP Package)

Normally fuse are set with **Factory specification** (unless specified otherwise)



Atmega328 micro-controller chip (QFP Package) mounted on the Arduino Uno board

fuse are set with **Arduino specification**

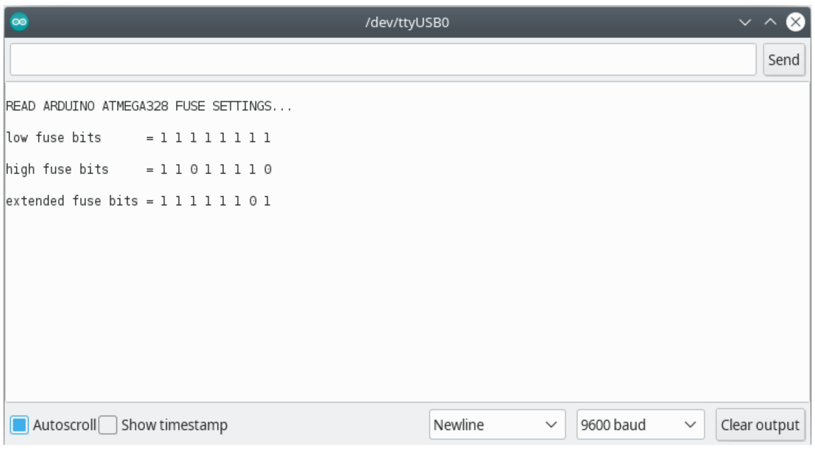
To know the exact setting values and their usage, please refer to the Atmega328 chip datasheet.

STEMKRAF – ATMEGA328P fuse bits
https://github.com/teaksoon/stemkraf

```
Program: stemkraf_read_atmega328_fuse
   (1/1): program to read fuse bit in ATMEGA328P
        :
        : by TeakSoon Ding for STEMKRAF (NOV-2021)
```

- Upload this program with the Arduino IDE Software
- Open up the Serial Monitor from the Arduino IDE Software
- Watch the Serial Monitor Screen

```
// Program: stemkraf_read_atmega328_fuse
//        : program to read atmega328 fuse
//        : WARNING!!! DO NOT ATTEMPT TO CHANGE the fuse setting if not sure
//        : by TeakSoon Ding for STEMKRAF ( NOV-2021 )
// ---------------------------------------------------
#include <avr/boot.h>

void setup() {
uint8_t low_fuse_bits, high_fuse_bits, extended_fuse_bits;

  low_fuse_bits       = boot_lock_fuse_bits_get(GET_LOW_FUSE_BITS);
  high_fuse_bits      = boot_lock_fuse_bits_get(GET_HIGH_FUSE_BITS);
  extended_fuse_bits  = boot_lock_fuse_bits_get(GET_EXTENDED_FUSE_BITS);

  Serial.begin(9600);
  Serial.print("\nREAD ARDUINO ATMEGA328 FUSE SETTINGS...");

  Serial.print("\n\nlow fuse bits      = ");
  Serial_monitor_show_bits(low_fuse_bits);

  Serial.print("\n\nhigh fuse bits     = ");
  Serial_monitor_show_bits(high_fuse_bits);

  Serial.print("\n\nextended fuse bits = ");
  Serial_monitor_show_bits(extended_fuse_bits);
}

void loop() {}

void serial_monitor_show_bits(uint8_t by) {
   for (int i=7; i>=0; i--) {
    Serial.print((by >> i) & 0x1);Serial.print(" ");
  }
}
```



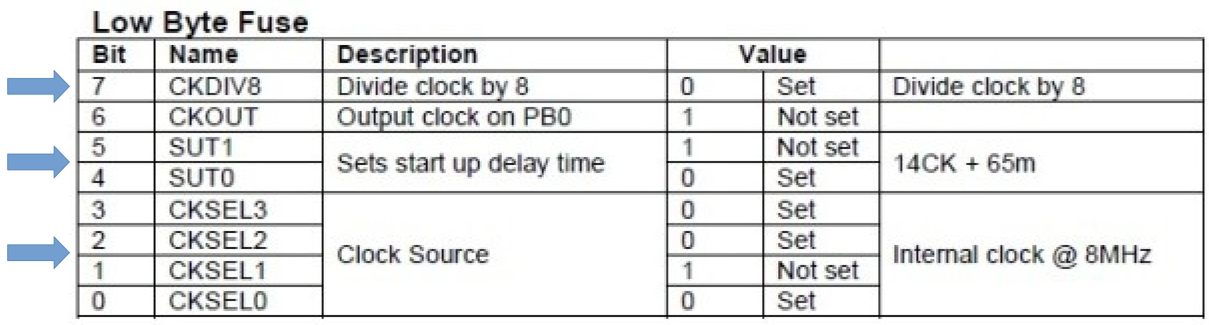
```
READ ARDUINO ATMEGA328 FUSE SETTINGS...

low fuse bits      = 1 1 1 1 1 1 1 1

high fuse bits     = 1 1 0 1 1 1 1 0

extended fuse bits = 1 1 1 1 1 1 0 1
```

This program only read the fuse bits from the ATMEGA328 chip and show it on the Serial Monitor Screen.

WARNING!!! Do not attempt to change the the default fuse settings if you are not sure. You may render the chip “unusable”

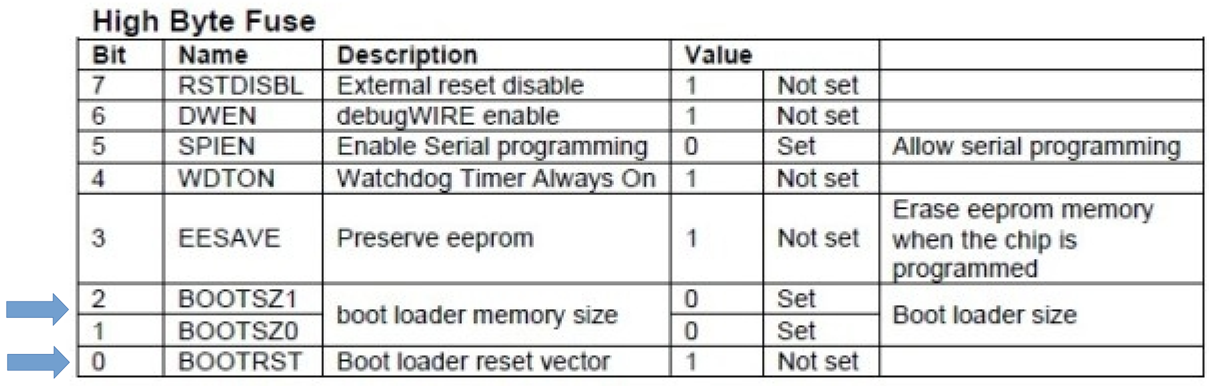
ATMEGA328 chip fuse settings, FACTORY setting vs ARDUINO setting (bit-7 ... bit-0)

```
Factory low fuse bits       = 0 1 1 0 0 0 1 0
Arduino low fuse bits       = 1 1 1 1 1 1 1 1

Factory high fuse bits      = 1 1 0 1 1 0 0 1
Arduino high fuse bits      = 1 1 0 1 1 1 1 0

Factory extended fuse bits = 1 1 1 1 1 1 1 1
Arduino extended fuse bits = 1 1 1 1 1 1 0 1
```



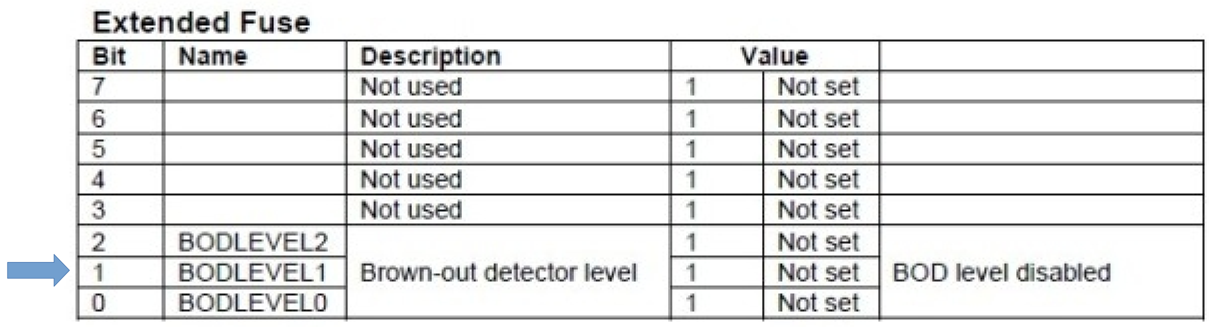
Low Byte Fuse

Bit	Name	Description	Value		
7	CKDIV8	Divide clock by 8	0	Set	Divide clock by 8
6	CKOUT	Output clock on PB0	1	Not set	
5	SUT1	Sets start up delay time	1	Not set	14CK + 65m
4	SUT0		0	Set	
3	CKSEL3	Clock Source	0	Set	Internal clock @ 8MHz
2	CKSEL2		0	Set	
1	CKSEL1		1	Not set	
0	CKSEL0		0	Set	



High Byte Fuse

Bit	Name	Description	Value		
7	RSTDISBL	External reset disable	1	Not set	
6	DWEN	debugWIRE enable	1	Not set	
5	SPIEN	Enable Serial programming	0	Set	Allow serial programming
4	WDTON	Watchdog Timer Always On	1	Not set	
3	EESAVE	Preserve eeprom	1	Not set	Erase eeprom memory when the chip is programmed
2	BOOTSZ1	boot loader memory size	0	Set	Boot loader size
1	BOOTSZ0		0	Set	
0	BOOTRST	Boot loader reset vector	1	Not set	



Extended Fuse

Bit	Name	Description	Value		
7		Not used	1	Not set	
6		Not used	1	Not set	
5		Not used	1	Not set	
4		Not used	1	Not set	
3		Not used	1	Not set	
2	BODLEVEL2	Brown-out detector level	1	Not set	BOD level disabled
1	BODLEVEL1		1	Not set	
0	BODLEVEL0		1	Not set	

The blank Factory Atmega328 chip and the Atmega328 chip used in Arduino Uno, Mini and Nano are physically the same. The difference are only in the following fuse settings.

CKSEL3,CKSEL2,CKSEL1,CKSEL0, type of Clock used (internal, external...)
SUT1,SUT0, Startup time delay. This setting depends on **CKSEL**
CKDIV8, clock Speed is divided by 8

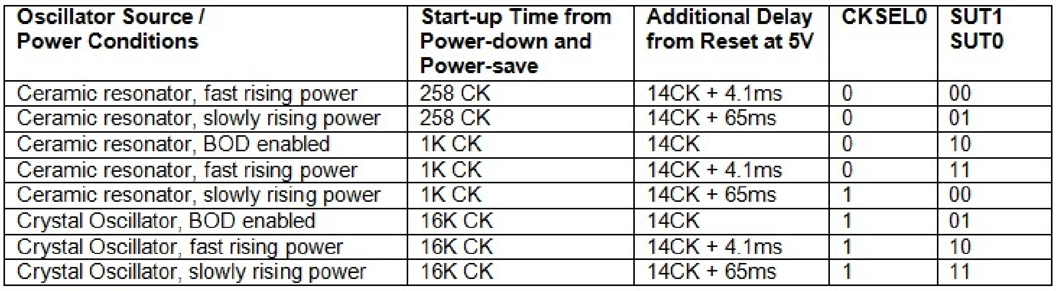
BOOTRST, determine whether “bootloader” program will be used or not
BOOTSZ1,BOOTSZ0, allocate Memory size for “bootloader” program

BODLEVEL3,BODLEVEL2,BODLEVEL10, determine level of BOD (Brown Out Detection, min voltage required)

CKSEL3,CKSEL2,CKSEL1,CKSEL0, type of Clock used (internal, external...)
CKSEL for **Factory fuse setting** is 0010, chip will **use chip internal 8Mhz Clock**
CKSEL for **Arduino fuse setting** is 1111, chip will **use external 16Mhz Clock** connected to chip pin XTAL1 and XTAL2. That is why we use the Atmega328 with Factory fuse setting, we do not need any external Clock Crystal BUT we must have an external Clock Crystal when we use the Atmega328with Arduino fuse setting

CKDIV8, Clock Speed is divided by 8
Factory fuse setting is enabled, the chip is running at **1/8 of the clock speed, 1Mhz** (8Mhz Clock divided by 8)
Arduino fuse setting is disabled, the chip is running at **full clock speed, 16Mhz** (16Mhz clock is not divided by 8)

SUT1 and SUT0, Startup time delay. This setting depends on the type of Clock used by **CKSEL**. Different Clock requires different time to stabilize before we use them.



Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset at 5V	CKSEL0	SUT1 SUT0
Ceramic resonator, fast rising power	258 CK	14CK + 4.1ms	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65ms	0	01
Ceramic resonator, BOD enabled	1K CK	14CK	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1ms	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65ms	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65ms	1	11

BOOTRST, determine whether a “bootloader” program will be used or not
Factory fuse setting is disabled, “bootloader” program is not required. When the chip is powered up, it will immediately run our main Program.
Arduino fuse setting is enabled, “bootloader” program is required along with our main Program. When the chip is powered up, it will run the “bootloader” program first, then only it will run our main Program.

Board like Arduino Uno, has a small C Language bootloader program named “optiboot” in its Atmega328 chip. “optiboot” deals mostly with setting up the the USB/TTL device (for program upload). You can replace “optiboot” with other program.

BOOTSZ1,BOOTSZ0, allocate Memory size for bootloader program
BOOTSZ1 and BOOTSZ0 **is not used for Factory fuse settings** since BOOTRST is diabled (they are simply set to 00).
BOOTSZ1 and BOOTSZ0 **in Arduino fuse setting is used to allocate allocate Memory** for the bootloader program (Memory taken from the Program Memory)

11 is 512 bytes, 10 is 1024 bytes, 01 is 2048 bytes and 00 is 4096 bytes. For the case of “optiboot” bootloader program, it is 11 since “optiboot” uses 500 bytes

BODLEVEL2,BODLEVEL1,BODLEVEL0, determine level of BOD (Brown Out Detection, min voltage required) - This is to prevent chip working in an under-voltage condition especially at higher speed. When there is not enough voltage, the chip will reboot). Since the Factory chip is running at only 1Mhz, any voltage will be fine, there is no need to set the BOD

The above just the differencs between the Factory fuse settings and Arduino fuse settings. There are a few other fuse that we can set but normally not required to do so.

Be extra careful with SPEIN, RSTDISBL and DWEN because they can potentially make your chip “unusable”. Read about them from the datasheet and you will understand why we NORMALLY WE WILL NOT TOUCH THEM.

IN SUMMARY

Normally we do not need to change the atmega328 chip fuse, we can use any atmega328 chip as it is.

Just choose either the Factory fuse setting chip or the Arduino fuse setting chip. These two choice gives enough options for most projects. We can still set the fuse when needed.

A. Use the Factory Fuse Setting Atmega328 chip in our project when we:

A1. Do not need a External Clock, we just want to use the 8Mhz internal clock and run it at 1Mhz.

Why would anyone wants to run at lower speed ? Lower speed uses less power, our device can operate longer time on a single battery charge

A2. Do not need to upload program via the USB cable from Arduino IDE Software, we can do it via SPI, therefore no need for a bootloader program.

A3. We need extra 512 bytes for our Program

B. Use the Arduino Fuse Setting Atmega328 chip in our project when we:

B1. We want to use External Clock, just like the 16Mhz Clock that Arduino Uno/Nano/Mini uses to run at 16Mhz speed

B2. We want to upload program via the USB cable from the Arduino IDE Software, requires the bootloader program, normally already pre-loaded (the pre-loaded bootloader can be replaced or we just use whatever that is available)

B3. We do not need extra 512 bytes for our Program

We can switch between the two, Arduino and Factory by just changing the fuse settings. Changing from Factory to Arduino requires another step, we need to install the “optiboot” bootloader program.

“optiboot” – https://github.com/Optiboot/optiboot/releases