Fuses and Lock Bits

The ATMega128 has several special memory areas

-Six lock bits which determine program memory features.
-Three fuse bytes which determine various hardware features
-Three bytes for device signature
-Four bytes for RC oscillator calibration (shared with device signature)

The six lock bits are broken down into:
-four BLB lock bits
-two lock bits for overall programming protection

The lock bits select where the *SPM* and *ELPM* are allowed to operate and if the bootloader is allow to overwrite itself.

Lock bits are logic zero when programmed, one when not programmed

Fuses and Lock Bits

The Fuse bytes are the
- -Extended Fuse Byte
- -Low Fuse Byte
- -High Fuse Byte

Extended Fuse Byte determines
- -M103C (mega 128 predesessor) compatibility
- -if watchdog Timer is always on

Table 117. Extended Fuse Byte

Extended Fuse Byte	Bit No.	Description	Default Value
–	7	–	1
–	6	–	1
–	5	–	1
–	4	–	1
–	3	–	1
–	2	–	1
M103C(1)	1	ATmega103 compatibility mode	0 (programmed)
WDTON(2)	0	Watchdog Timer always on	1 (unprogrammed)

Notes: 1. See "ATmega103 and ATmega128 Compatibility" on page 4 for details.
2. See "Watchdog Timer Control Register – WDTCR" on page 56 for details.

Fuses and Lock Bits

The Fuse bytes are the
- -Extended Fuse Byte
- -Low Fuse Byte
- -High Fuse Byte

Extended Fuse Byte determines
- -M103C (mega 128 predesessor) compatibility
- -if watchdog Timer is always on

High Fuse Byte determines
- -availability of OCD, JTAG, SPI downloading
- -if oscillator options are available
- -if EEPROM is preserved during chip erase
- -boot block size
- -if reset vector is application or bootloader address zero

Fuse bits are logic zero when programmed, logic one when not programmed
Fuse bits are inaccessible by application or bootloader programs
Fuse bits not effected by chip erase or at all if lock bit (LB1) is set

Fuses and Lock Bits

High Fuse Byte determines

-availability of OCD, JTAG, SPI downloading
-if oscillator options are available
-if EEPROM is preserved during chip erase
-boot block size
-if reset vector is application or bootloader address zero

Table 118. Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN[4]	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN[5]	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN[1]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[2]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 112 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 112 for details)	0 (programmed)[3]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Notes:
1. The SPIEN fuse is not accessible in SPI Serial Programming mode.
2. The CKOPT fuse functionality depends on the setting of the CKSEL bits. See "Clock Sources" on page 37 for details.
3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 112 on page 284
4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of lock bits and the JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.
5. If the JTAG interface is left unconnected, the JTAGEN fuse should if possible be disabled. This to avoid static current at the TDO pin in the JTAG interface.

Fuses and Lock Bits

Low Fuse Byte determines

-brownout detector trigger level

-startup time

-clock sources

Table 119. Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown out detector trigger level	1 (unprogrammed)
BODEN	6	Brown out detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

Notes: 1. The default value of SUT1..0 results in maximum start-up time. See Table 14 on page 42 for details.

2. The default setting of CKSEL3..0 results in Internal RC Oscillator @ 1 MHz. See Table 6 on page 37 for details.

Fuse bits are logic zero when programmed, logic one when not programmed

Fuse bits are inaccessible by application or bootloader programs

Fuse bits not effected by chip erase or at all if lock bit (LB1) is set

Fuses and Lock Bits

Fuse bytes can be read/written with avrdude interactively in terminal mode

```
avrdude -p m128 -u -c usbasp -t  //unsafe option chosen (see manual)

avrdude: AVR device initialized and ready to accept instructions
avrdude: Device signature = 0x1e9702
avrdude: current erase-rewrite cycle count is 52 (if being tracked)
avrdude> d efuse
>>> d efuse                 //dump entended fuse
0000  fd

avrdude> d hfuse            //dump high fuse
>>> d hfuse
0000  99

avrdude> d lfuse            //dump low fuse
>>> d lfuse
0000  e1

avrdude> w efuse 0 0xff     //write extended fuse
>>> w efuse 0 0xff

avrdude> w hfuse 0 0x89     //wire high fuse
>>> w hfuse 0 0x89

avrdude> w lfuse 0 0x2f     //write low fuse
>>> w lfuse 0 0x2f
```

Fuses and Lock Bits

Fuse bytes (and memory) can be read/written with avrdude from the command line.

For example, when programming a Mega48 to run at 8Mhz, the low fuse needs to be changed to `0xE2`. It can be done like this:

```
sudo avrdude -c usbasp -p m48 -U lfuse:w:0xE2:m -v

"-U"     indicates a memory operation,
"w"      indicates a write
"lfuse"  the location to perform the operation on
"0xe2"   the hex value being written
"m"      take the immediate value specified on command line
"-v"     verbose mode
```

Multiple operations can be done simultaneously too:

```
avrdude -p m128 -u -U flash:w:diag.hex    \
>                  -U eeprom:w:eeprom.hex \
>                  -U efuse:w:0xff:m      \
>                  -U hfuse:w:0x89:m      \
>                  -U lfuse:w:0x2e:m
```

Verifying from command line that flash memory is same as the file **`code.hex`**:

```
avrdude -p m128 -c <programmer_type> -U flash:v:code.hex    \
```

See the avrdude manual for more details:

`http://www.nongnu.org/avrdude/user-manual/avrdude.html`