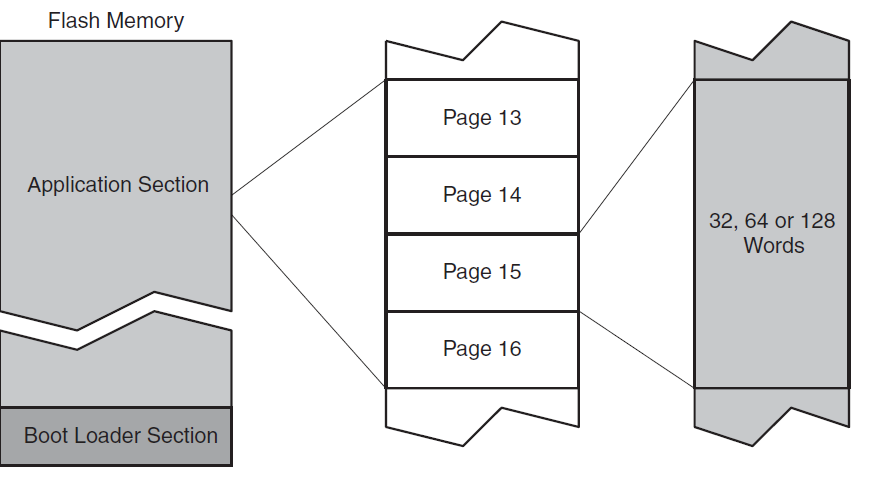
Bootloader

A bootloader is just a piece of code that is the first to execute when the processor turns on before the main program executes, hence the name, it boots the device. On a microcontroller this is used for programming the chip.

The AVR platform supports this and calls it *Self Programming* which is supported on most chips. The bootloader is placed in a special section of the flash memory.

# Memory Organization

All memory is divided into pages containing 32, 64 or 128 words each. Paging is a memory management technique in which memory is divided in fixed pages.



The flash memory is divided into two sections, the Application section and the Boot Loader section. As the names suggests the Application section contains the main code of the application, while the Boot Loader section contains code for the actual bootloading.

The size of the bootloader is configurable through two *BOOTSZx* fuses. Look into the datasheet for details.

If a bootloader is implemented, it can be called from the Application code using calls or jumps, or by programming the *BOOTRST* fuse. When this fuse is programmed, the CPU starts execution in the Boot Loader section on reset (instead of starting at 0x0000.)

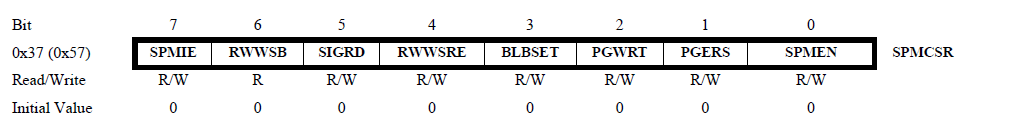
The SPM instruction can only be executed from the Boot Loader section which residents in a special region called the NRWW section.

# Read-While-Write

In addition of the division between Application and Boot Loader sections, the flash also divides into two fixed-size sections. The first section is the Read-While-Write (RWW) section. The other is the No-Read-While-Write (NRWW) section. The NRWW section always equals the largest selectable Boot Loader size, for the ATmega328p this is 2048 words (or 4096 bytes.)

The use of these sections is that the NRWW section is accessible while updating the RWW section. It is impossible to access the RWW section while its updated. Thus No-Read-While-Writing to the NRWW section, but possible to Read-While-Writing to the RWW section.

# SPM Instruction

All Self-programming operations are performed by the SPM (Store Program Memory) instruction. SPM can be used to erase and write pages, fill the Page Buffer, set the Boot Loader lock bits. When using the SPM instruction, the SPMEN bit must always be set within four cycles prior executing the SPM instruction. How the instruction behaves is selected through the SPMC(S)R register.

When the SPMEN bit is set with either RWWSRE, BLBSET, PGWRT or PGERS the following SPM instruction will have a special meaning.

### RWWSRE: Read-While-Write Section Read Enable

When programming to the RWW section, the RWW section is blocked for reading. To re-enable the RWW section, the software must wait until programming is completed. Then, if RWWSRE bit is written the next SPM instruction re-enables the RWW section.

### BLBSET: Boot Lock Set Bit

The next SPM instruction causes the Boot Lock and Memory Lock bits to be set according to the data in R0.

### PGWRT: Page Write

The next SPM instruction executes a Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The CPU is halted during the entire Page Write if the NRWW section is addressed.

### PGERS: Page Erase

The next SPM instruction executes a Page Erase. The address is taken from the high part of the Z-pointer. The CPU is halted during the entire Page Write if the NRWW section is addressed.

# Temporary Page

1. All Flash updates are done page by page. Before writing to a page, the page must be erased. The page is addressed through the upper bits of the Z-register.
2. To write new data to a page, the Page Buffer must be filled first. The page buffer is a separate write-only buffer holding one temporary page. This buffer must be filled word by word. To write a word to the Page Buffer, load the word into the R1:R0 registers, set the Z-register to point to the correct word.
3. When the Page buffer is filled it must be written to Flash memory. Set the high bits of the Z-register to address the page.

The SPMEN bit can be polled to find out if the CPU is ready. When performing a Page Erase or Write on the RWW section the RWWSB Flag is set by hardware. This indicates that the section is inaccessible. The flag should be cleared by software when the SPM is completed using the RWWSRE bit. If RWW is accessed without re-enabling all addresses read 0xFFFF. The consequence of this is that when jumping to this section (i.e. to resume the main program) the code 0xFFFF is executed, which it will just falls through it finds executable code which is the first address of the NRWW section.

# Intel HEX

The output of avr-objcopy is a .hex file. Intel HEX is a file format that conveys binary information in ASCII text form. It is commonly used for programming microcontrollers. The file is imported by a programmer to program the machine code to the device or the file is transferred to target and loaded there.

Intel HEX consists of lines of ASCII text that are separated by a line feed or carriage return. The binary data is encoded in hexadecimal characters. Each line is called a record.

## Record

A record consists of six fields:

1. Start code, one character, an ASCII colon ‘:’
2. Byte count, two hex digits, indicating the length of the data field.
3. Address, four hex digits, representing the 16-bit beginning memory address offset of the data. The physical address is computed by adding this offset to a base address.
4. Record type, two hex digits, defining the data.
5. Data, a sequence of n bytes, represented by 2n hex digits.
6. Checksum, two hex digits, used for validating the record.

# References

|  |  |
| --- | --- |
| AVR109: Self Programming | <http://www.atmel.com/images/doc1644.pdf> |
| ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet | <http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf> |
| Atmel AVR 8-bit Instruction Set | <http://www.atmel.com/images/atmel-0856-avr-instruction-set-manual.pdf> |
| Intel HEX | <https://en.wikipedia.org/wiki/Intel_HEX> |
| AVR Bootloader FAQ | <http://www.avrfreaks.net/sites/default/files/bootloader_faq.pdf> |