Application Note
Loading Bootblock Code with Kanda In System and Keyfob Programmers

Introduction

Bootblock and Application memory are both in the Flash memory of the AVR. If no Bootloader code is required, then the whole Flash memory is used for user application code as normal and the subject can be ignored.

A separate Bootblock loader can be used to reprogram the Flash memory using the LPM and SPM commands (Load Program Memory/Store Program Memory). The new code can be sent to the microcontroller via any communication method e.g. UART or SPI, which gives flexibility.

This note covers how to program bootloader code into the AVR using the serial programming interface used by Kanda programmers. It is not intended to cover the details of Bootloader code, which is described fully in AVR Device datasheets.

Bootloader Basics

The Bootloader section is at the end of the Flash memory. The size of the Bootloader section can be changed by altering the number of pages but the number of pages that can be selected is device dependent. The page size is also device dependent e.g. ATmega8 has page sizes of 128 words and ATmega128 has page sizes of 512 words. Therefore, the actual maximum size of the Bootloader Block varies from device to device (1028 to 4096 words).

As the Bootloader block is at the end of the Flash memory, its **Start Address** will depend on :
The number of pages selected
The page size of the device
The overall Flash memory size of the device.

A table is given in each device datasheet called **Boot Size Configuration**. This table includes the **Start Address** for the Bootloader section for different sizes of Boot Block on that particular device. The Start Address is also called the **Bootloader Reset Address** or **Reset Vector**.

Including Bootloader code

The programmer needs a single file to be loaded into the Flash memory buffer. So this file must include both your Application code and any Bootloader code which must originate at the **Start Address**.

The simplest way of achieving this is to write the Bootloader code as a separate file and assemble it to determine its final size in words. Once the size is known, use the **Boot Size Configuration Table** in the datasheet of your chosen device to choose the correct size Bootloader block and its **Start Address.**
Once you have the correct **Start Address,** add an ORG directive to the end of your application code assember file to set the Start Address for the Bootloader code. The syntax will vary depending on your chosen assember (probably ORG or .ORG).
Now include the bootloader file using the Include directive (.Include or

#Include) and assemble the file. You will now have one Flash Memory file that can be loaded into the programmer using the **File-Load-Flash Memory** command.
Alternatively, you can add the ORG [Start Address] directive to the top of the Bootloader assembler file and user a linker.

Fuses

After loading the Flash memory buffer with the file containg both Bootloader and Application code - **File-Load-Flash Memory** - click on the **Fuses and Lockbits** tab. Now select the **Lockbits and Boot Options** tab at the side of the Fuses and Lockbits window.

Use the **Boot Block Size** drop down list to set the size of the Boot Block to your calculated size. Note that each device has different size options.

Set the **Reset Vector** to either Application or Boot Block depending on which block of code you want to run after device reset. If you set the Reset Vector to Boot Block, your Application will not run as the code will start executing from the Start Address of the Bootloader Block.

Normally, you would set the Reset Vector to Application and use a jump to the Boot Block when a command is received on SPI, UART or other communication method.
If no Bootloader is required, just ensure that the Reset Vector is set to Application.

Lock bits

There are three security sets on ATmega AVR devices:

Lockbits - These are the normal Lockbits for setting access to the whole Flash memory and are used in the normal way - Not set, No further writes, No read or write

BLB0 - (Boot Lock Bits 0) These, if programmed, prevent SPM/LPM access to the Application code area of the Flash Memory.

BLB1 - (Boot Lock Bits 1) These, if programmed, prevent SPM/LPM access to the Boot Block area of the Flash Memory.

All these lockbits should be set if you require complete security for your code as LPM/SPM instructions can be used to read your Bootloader or Application code even if the main Lockbits are set. The only way to clear lockbits is to use the Device-Erase command.
For further security you can encrypt your Bootloader code.

Kanda.com

Embedded Results Ltd
P.O. Box 200
Aberystwyth,
SY23 2WD UK

Tel: +44 (0) 1974 261 273
Fax: +44 (0) 1974 261 273
Email: sales@kanda.com
Web: www.kanda.com