Atmega 328 Bootloader for hex and text plus calibrator for the internal clock: User guide

Postings have already been made for

A text programmer that places text files in flash below the hex file
A bootloader that programs applications
Calibration routines for the internal RC oscillator

In this posting firmware is presented that provides all three of these functions. A bootloader for both hex and text files occupies the space in the target device between addresses 0x7000 and 0x7FFF. It programs it own flash at addresses 0x6C7F and below with text, and at addresses zero and above with an application. The space between 0x6C7F and 0x6FFF is programmed with an auto calibration routine.

The SW supplied

All source files are provided for:

328 Bootloader for hex&text	A combined hex/text bootloader
328_autoCal_subroutine	A calibrator for the internal 8MHz RC clock
Short_Application	A basic application required to read the text
Cal_disrupter	This disrupts the calibration of the RC clock.

Additional files include:

The hex file consisting of the combined bootloader and calibrator is also provided.
Batch files enabling the USBasp programmer to program Atmega 328 and 328P devices.

The Configuration bytes

The bootloader runs under the following configuration bytes:

Fuse	Value	Configuration setting
Extended	0xFD/5	2.7V Brown out detector
High	0xD0	Enables the external reset, puts the WDT under program control Preserves the EEPROM through chip erase Maximum boot size, Resets to the boot partition
Low	0xC2	Minimum start up timer for use with BOD 8MHz internal RC clock
Lock	0xEF/2F	Bootloader section can only be programmed externally

Both the text programmer and user applications will have to operate under the same configuration bytes, however the only settings that are critical to the bootloader are:

External reset enabled
WDT under program control
Maximum boot size
Reset to start of bootloader section.
EEPROM preservation

If the device does not use the internal clock then the calibration routine will never be used and its presence can be ignored.

Generating the bootloader/calibrator hex file

When the bootloader is compiled using the makefile supplied the last line but one is:

:040000030000700089

This is loaded at location zero and causes the program counter to jump to address 0x7000.
This line serves no purpose and can be deleted from the hex file.

Similarly the last two lines of the autoCal routine which are

:0400000300006C800D This causes a jump from zero to 0x6C80
:00000001FF This signals the end of the hex file

should also be deleted.

When to two hex files have been amended as above the bootloader hex file should be appended to the bottom of the autoCal hex file and the combined file should be used to program the target Atmega 328/P device.

Running the autoCal routine.

When the bootloader is first used after programming the following user prompt should be generated:

h/t/r h/t/r h/t/r h/t/r……

However if the default OSCCAL value supplied with the chip is not sufficiently accurate the user prompt will not be intelligible.
The autoCal routine should therefore be used. It generates user calibration values that will automatically be saved to EEPROM for use following all futures power on resets. To do this hold the device in reset while power is applied. The bootloader will then execute a jump to the start of the autoCal routine which runs and ends with a SW_reset. The h/t/r user prompt should now be intelligible.

Format of the text file

The text file can be created using a word processor provided that the completed version is saved as a text file:
The text file contains three sections:

The first, before a row of * characters contains an optional introduction to the data
The second immediately after the row of * characters contains optional carriage returns
The third contains the text strings to be saved to flash (which may contain additional * characters).

Program operation

A commercial programmer such as the USBasp may be used to program the target device with the combined bootloader/calibrator hex file. It can then be transferred to a pcb and connected to a PC via a USB port. Following a reset the bootloader generates the user prompt which can be viewed on a terminal program such as Br@y or Tera Term. Once the autoCal routine has, if needed been launched, the bootloader can be used:

Each of the three keypresses results in the following actions:

‘t’ A text file is requested and can be sent
‘h’ An application is requested and can be sent
‘r’ Control is passed from the bootloader to the application

Assuming that the application is the “Atmega328_app” supplied with this posting the user response changes to w/s w/s w/s……

Keypress ‘w’ resulted in the text being read out.

Following a reset, control is passed back to the bootloader. Keypress ‘L’ followed by ‘Y’ results in the device being locked. No further programming is immediately possible using the bootloader. The device must first of all be reprogrammed with the USBasp to clear the lock byte. It was found necessary to reprogram the flash before the lock byte could be reset.

Note: The only reason that it has been found possible to combine the hex and text programmers is that the verification subroutines have been omitted. It has been found that in general the bootloader either works all the time or none of it. Verification was vital during program development but can really be dispensed with afterwards.

Disrupting the calibration

The user prompt for the “Cal_disrupter” routine is

“w-Press W to disrupt the calibration/stuvwxyz”

Following the ‘w’ keypress the default value of OSCCAL is incremented by 15 and saved to EEPROM.
No communication between the device and PC is therefore possible until the “autoCal_subroutine” has been launched.

Hardware

The hardware will depend on the requirements of the user application. An external reset is required however which will pass control to the bootloader. All other resets will pass control to location zero. An activity led is assumed on ports B0 but this can be omitted.

Program development

WinAVR-20100110 was used exclusively during program development. The GUI consisted of Programmers notepad and a terminal program such as Tera Term or Br@y++.