Text programmer for the Atmega 328:   User guide

The SW supplied

All C files, an assembly file and makefile are provided for both the text programmer, and also for a simple application. This is intended to show how the text programmer has been designed to work with user applications. The hex files are also provided which were generated using WinAVR-20100110.

The text programmer hex file starts at address 0x7000 and easily fits into the space available for it, so there is scope for further developments.

The Configuration bytes

The text programmer runs under the following configuration bytes:

Fuse	Value	Configuration setting
Extended	0xFD	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	Bootloader section can only be programmed externally

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

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

Generating the application/text programmer hex file

When the text programmer 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 should be deleted from the hex file, otherwise it will corrupt the first line of the application.
Similarly the last line of the application which is :00000001FF should be deleted because it signals the end of the hex file.
When to two hex files have been amended as above the text programmer hex file should be appended to the bottom of the application hex file and the combined file should be used to program the target Atmega 328 device.

Programming the flash with text

At the point in time that the target device is released by the programmer no text has been programmed to the flash and the “text programmer” runs.

It generates the following user prompt: “? ? ? ? ?……...”
If the user responds with keypress “V” the SW version is send to the PC.
If “?” is pressed the following message is sent:
    “Send and then echo file”
The user sends the text file and the text programmer echo’s it to the screen.
The user then responds with “Y” or “N” to either accept or resend the file.
Assuming that “Y” is eventually selected control then jumps to the application program.

In this case the user prompt is “w/s   w/s   w/s………………...”
A “w” keypress results in the number of strings being displayed.
The user then enters the string number i.e. 5 followed by carriage return and the fifth string is printed out.

Note: If the target device is power cycled before being programmed with text, the text programmer will run so that text may be programmed.

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).

The space available for text

Text starts filling flash at location 0x6FFFF and can occupy as much space as the application, which starts at location zero allows. Between them they have space for 0x7000 bytes (> 28 K bytes). This compares with 1 K byte for text stored in EEPROM or <<2 K bytes for text embedded in the C-program that will automatically be copied to SRAM at run time.

Running the user application

When the target device is power cycled or reset in any way the user application will run immediately because text has already been programmed to the flash. The reset vector is still location 0x7000 but a jump to location zero is executed almost immediately.

Additional background information
This is given in a sample text file that is supposed to be an example of a typical user text file.

Hardware

The text programmer does not require any external hardware. Even the external reset required by the bootloader programmer upon which it is based is no longer required. No hardware details or circuit diagrams are therefore given. Users might optionally want to add a flashing LED to indicate that file download is successfully taking place. The header file assumes that PB0 is used for this purpose.