
AVR1605: XMEGA™ Boot Loader Quick Start Guide

Features

- XMEGA boot loader
- AVROSP compatible
- Example application
- C-code sample application for Self Programming
- Read and Write Both Flash and EEPROM Memories
- Read and Write Lock Bits
- Read Fuse Bits

1 Introduction

This application note describes how to use a boot loader application with one of the XMEGA family devices (i.e. ATxmega128A1) and how an AVR® with the Store Program Memory (SPM) instruction can be configured for Self-programming. The sample application communicates via the UART with a PC running the AVR Open Source Programmer (AVROSP) from Application note AVR911. This enables the user to download Applications into Flash, Data to EEPROM and to read/write fuses without the need for an external programmer. The example software is using the ATxmega128A1 device with STK®600 as a target board.

Electronic designs including a microcontroller always need to be equipped with a firmware, be it a portable music player, a hairdryer or a sewing machine. As many Electronic designs evolve rapidly there is a growing need for being able to update Products, which have already been shipped or sold. It may prove difficult to make changes to the hardware, especially if the product has already reached the end customer. But the firmware can easily be updated on products based on Flash microcontrollers, such as the AVR family.

Many AVR microcontrollers are configured such that it is possible to create a boot loader able to receive firmware updates and to reprogram the Flash memory on demand. The program memory space is divided in two sections: the Boot loader Section (BLS) and the Application Section. A Boot Loader program is placed inside the Boot Section of the Flash memory. This program handles communication with the host PC, and facilitates programming of both Flash and EEPROM. Once programmed, different levels of protection can be individually applied to both the boot and application portion of the Flash memory. The AVR thus offers a unique flexibility, allowing the user extensive degrees of memory protection.

For general information about self programming please refer to application note AVR109: Self programming.



8-bit **AVR**[®]
Microcontrollers

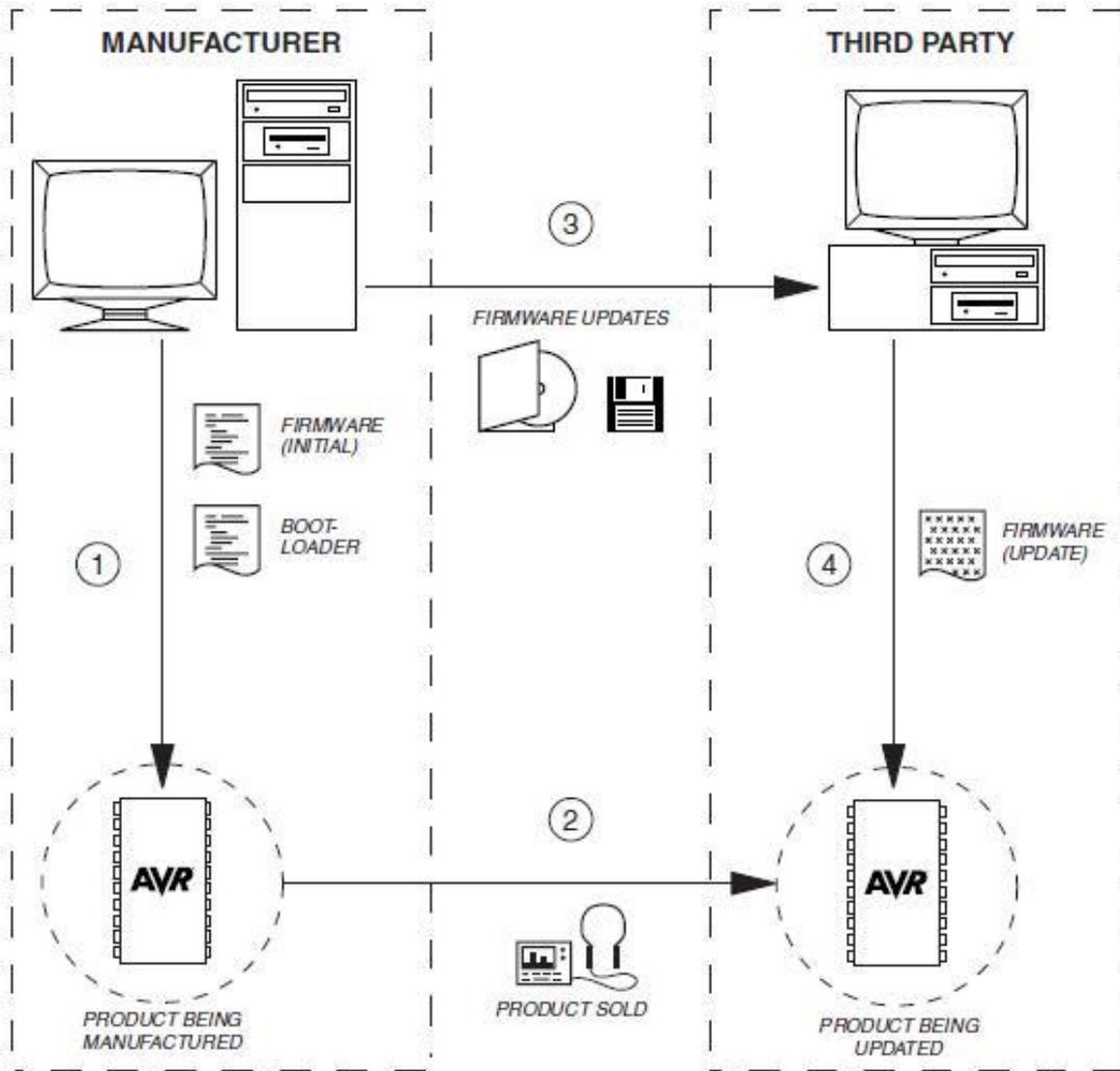
Application Note

Preliminary

Rev. 8242A-AVR-05/09



Figure 1-1. Purpose of Bootloader



2 Getting up and running

This section walks you through the basic steps for getting up and running, by setting up the hardware. The necessary setup and requirements are described along with relevant information.

2.1 Hardware Setup

This section gives the step by step procedure to setup the hardware for the experiment.

The Boot Loader software presented in this application note uses the AVR Open Source Programmer (AVROSP) as the user interface. The example application implements functions to read or update the Flash and EEPROM memories on the target device. It is also possible to read and update the Lock bits and read the Fuse bits of the device using the ATxmega128A1 AVR device with STK600.

On the STK600, connect the PD2/PD3 pins to the RXD/TXD pins to route the USART signals to the RS-232 output.

The STK600 has got two RS232 ports marked CAN (Male Type) and RS232 (Female Type). Connect a serial cable between the PC and the RS232 (Female Type) port of STK600.

On STK600, connect the PD4 pin to the SW0 pin to use micro switch SW0 to enable boot loader mode.

Please refer STK600 User Guide, available in AVR Studio[®] Help to connect it with PC and to mount the device with correct routing and socket cards combination.

2.2 System Configuration

This section gives the information about setting up your PC prior to start the implementation.

Include the xml_dev_files directory path (found in code\AVROSP_Test) to the windows PATH by creating a variable name and value. This is set because the system will search the default locations for the AVROSP.exe file to run from any location of the directory. If this is not done, then you can use AVROSP.exe in the folder where it is and you can not call it anywhere else. It can be set by right clicking My Computer icon and selecting the Properties option as demonstrated in the Figure 2.2-1, 2, 3 below. (You may need to restart the computer to enable the changes).

Figure 2.2-1. PATH Configuration 1

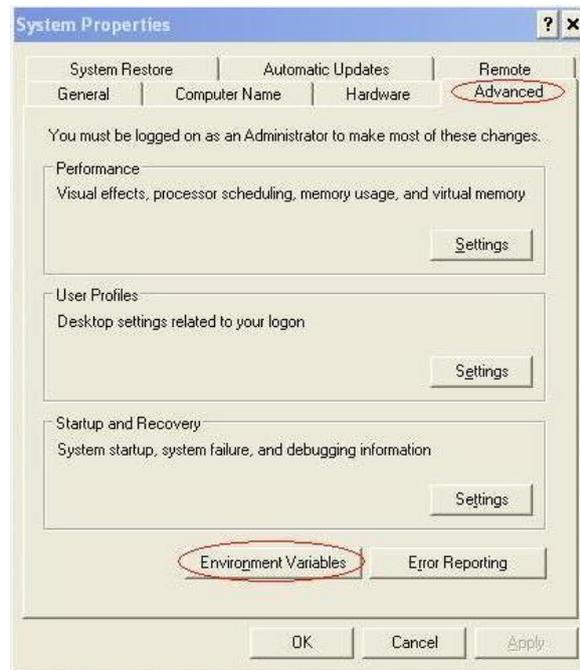


Figure 2.2-2. PATH Configuration 2

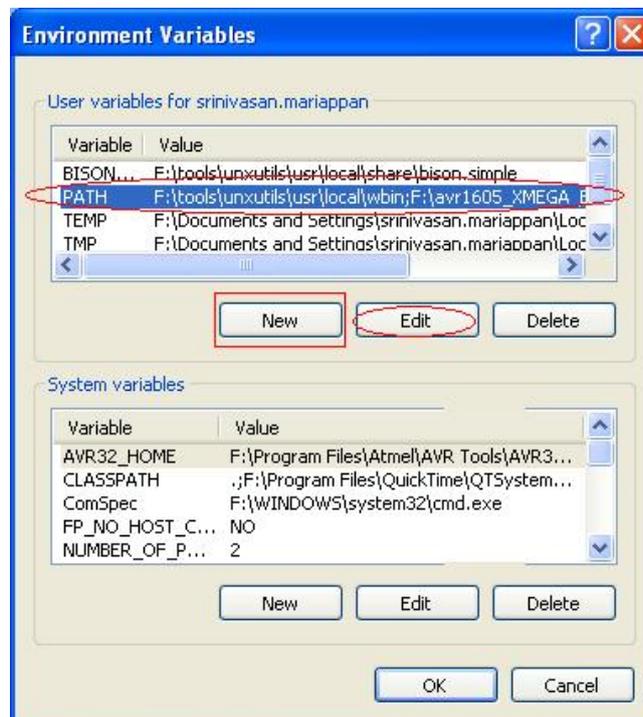


Figure 2.2-3. PATH Configuration 3



2.3 Starting a debug session

This section gives the step by step procedure to start the debugging session on the device.

Start IAR™ (v5.12C or later) and select the option "Open existing workspace". Browse for the folder downloaded along with this Application note (hereafter called as target folder) and select bootloader.eww for debugging. Verify that there are no errors or warnings by build and compilation.

Start AVR Studio and create a new project using bootldr.dbg. Select the device as ATxmega128A1 and the platform as JTAGICE mkII.

Make sure that both the STK600 and JTAGICE mkII is Powered ON.

In AVR studio open the programming dialog and connect to the ATxmega128A1 using the JTAG or the PDI interface. Select the fuses tab and set the BOOTRST fuse to Boot loader Reset and program the fuses.

Start a debugger session in AVR Studio and run the application while keeping SW0 pressed on the STK600 (Check the LED status of the Tool).

In Programming mode, the program receives commands from AVROSP via the UART. Each command executes an associated task. Any command not recognized by the boot loader program results in a "?" being sent back to AVROSP.

Through explorer, open the folder AVROSP_Test where several command line examples, example hex files for FLASH and EEPROM, the AVROSP.exe and the XML part description file for the ATxmega128A1 are available.

NOTE: Default PC's RS232 port is set to COM1, if the STK600 is connected to another port the following batch programs need to be modified. The batch files are as follows:

```
x128A1_chip_erase.bat
x128A1_eeprom_dump.bat
x128A1_eeprom_write_file.bat
x128A1_flash_dump.bat
x128A1_flash_write_file.bat
```

In all the batch files the port name COM1 in the first line have to be modified/replaced with COMx. Where x is the RS232 port number where the STK600 is connected.

The first line of the batch file is shown below which can be viewed by drag-drop of batch file into notepad application:





The following command will set up the COM1 in Windows when given in MSDOS command line:

```
mode com1 Data=8 Parity=n Baud=9600 DTR=OFF RTS=OFF to=off
```

3 Communication with the bootloader

This section walks you through the basic steps for communicating with the bootloader through the hardware setup. The necessary setup and requirements are described along with relevant information.

3.1 Chip Erase

This section explains how to implement the 'Chip Erase' command.

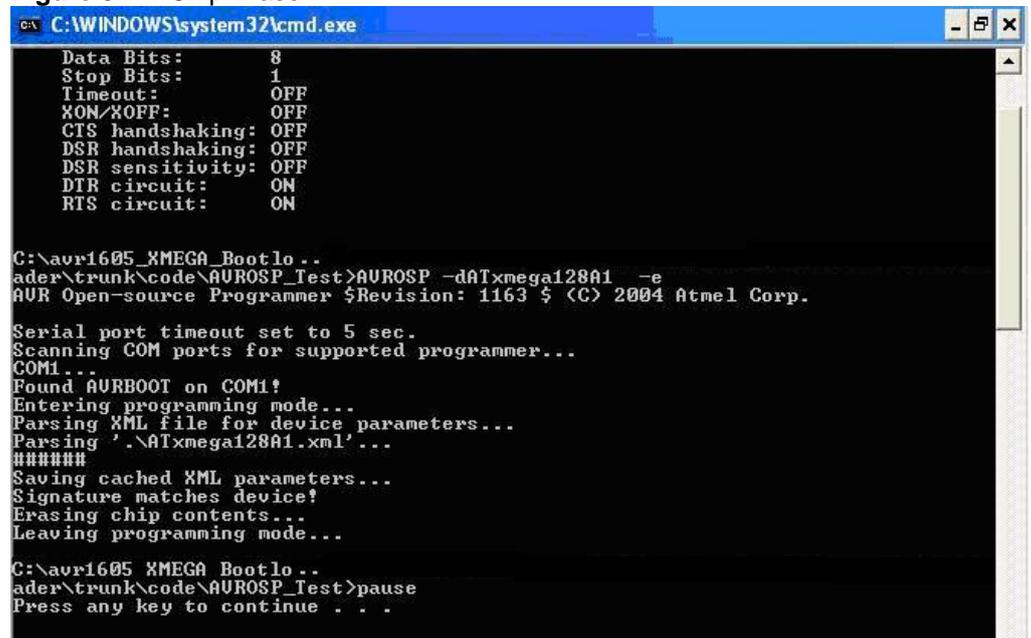
Ensure that the steps to make the device wait in bootloader section are done [Press both External RESET and SW0 on STK600 and release the RESET first and then the SW0].

In the current working folder, double-click (run) the batch program called x128A1_chip_erase.bat

The batch file runs the following command to erase the flash and the EEPROM when using AVROSP:

```
AVROSP -dATxmega128A1 -e
```

Figure 3.1-1 Chip Erase



The command window shown in Figure 3.1-1. Chip Erase should appear: After connecting to COM1, the device specific file is parsed for the ATxmega128A1 by AVROSP and the output file is placed in the AVROSP_Test directory. Further, the signature is matched and a chip erase is performed by the following operations:

- 1) Given COM port number is scanned

- 2) Communicates with AVRBOOT
- 3) Enters programming mode
- 4) XML file is parsed and new XML file is generated in the current working folder
- 5) Signature is checked to match
- 6) Chip erase is performed
- 7) Leaves the programming mode

3.2 Write a file to flash

This section explains how to write/download a file to the flash.

Ensure that the steps to make the device wait in bootloader section are done [Press both External RESET and SW0 buttons on STK600 and release the RESET first and then the SW0].

In windows explorer, run the batch file x128A1_flash_write_file.bat. The batch file runs the following command to program and verify the flash memory:

```
AVROSP -dATxmega128A1 -e -iffflash.hex -pf -vf
```

Table 3.2 -1. Command Description

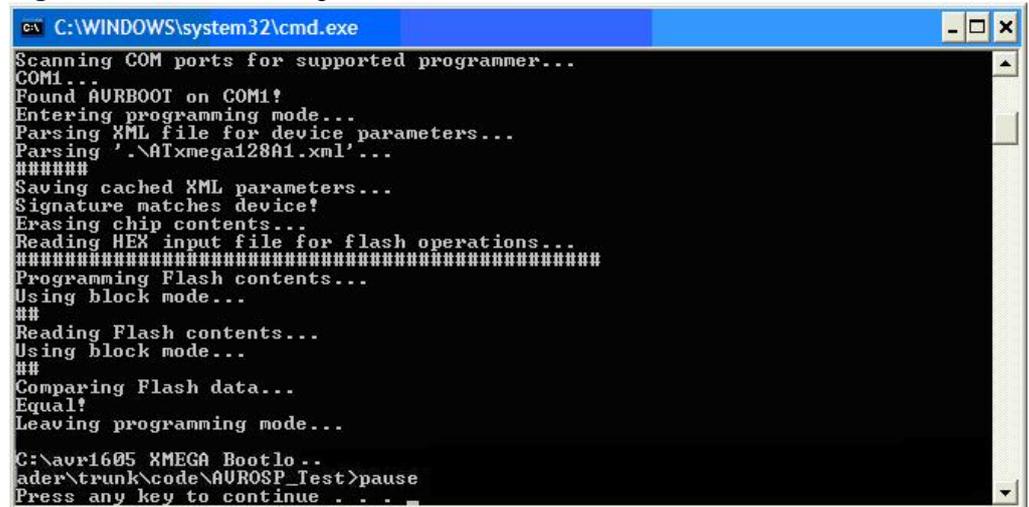
Command	Description
-e	This command will erase the flash the device.
-iffflash.hex	This command will give the 'flash.hex' file as an input file for programming the flash
-pf	This command will program the flash
-vf	This command will verify the flash content after programming

After comparing, by the verification command, "Equal!" string in the command window is to be noted which ensures that the programming of flash memory with the given input file is performed without any error.

Refer Figure 3.2-1 Downloading to Flash where you can see the "Equal!" string.



Figure 3.2-1. Downloading to Flash



3.3 Run the uploaded application from address 0

The flash.hex is a program that writes the string “Congratulations!” when it gets the character ‘E’ through serial/RS232 port.

The program will jump to the application section after RESET, provided the micro switch SW0 in not pressed at the time of RESET. Alternatively, the application can be run by setting the BOOTRST fuse to Application Boot and restart the part. (Note that a start of another debugging session will perform a chip erase).

Now the application will wait for the letter ‘E’ through USART0. To verify the application that was downloaded to the flash in the previous chapter, we must write the character ‘E’ through terminal software (e.g. HyperTerminal) as described below:

Open HyperTerminal program available in all the Windows based system by following the path Start→All Programs→Accessories→Communications→Hyper Terminal

Give any name(eg:Test) for that session and configure the COM port settings to 9600 baud, 8bit Data, no parity, 1 stop bit and no Handshaking. Please refer Figure 3.3-1.Hyperterminal Name and Figure 3.3-2.COM port Settings for more information.

Figure 3.3-1 HyperTerminal Name

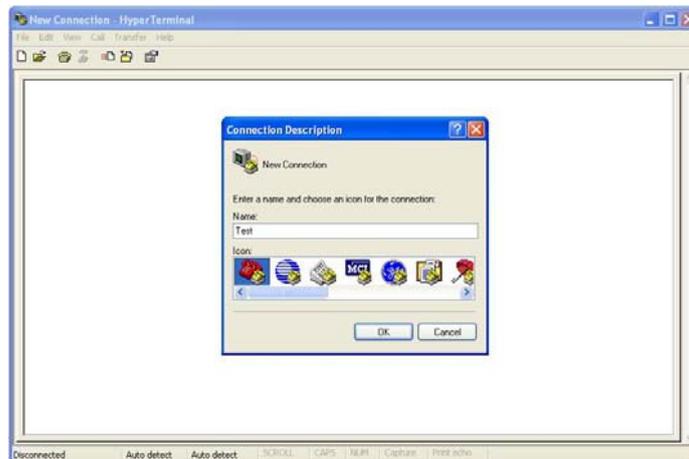
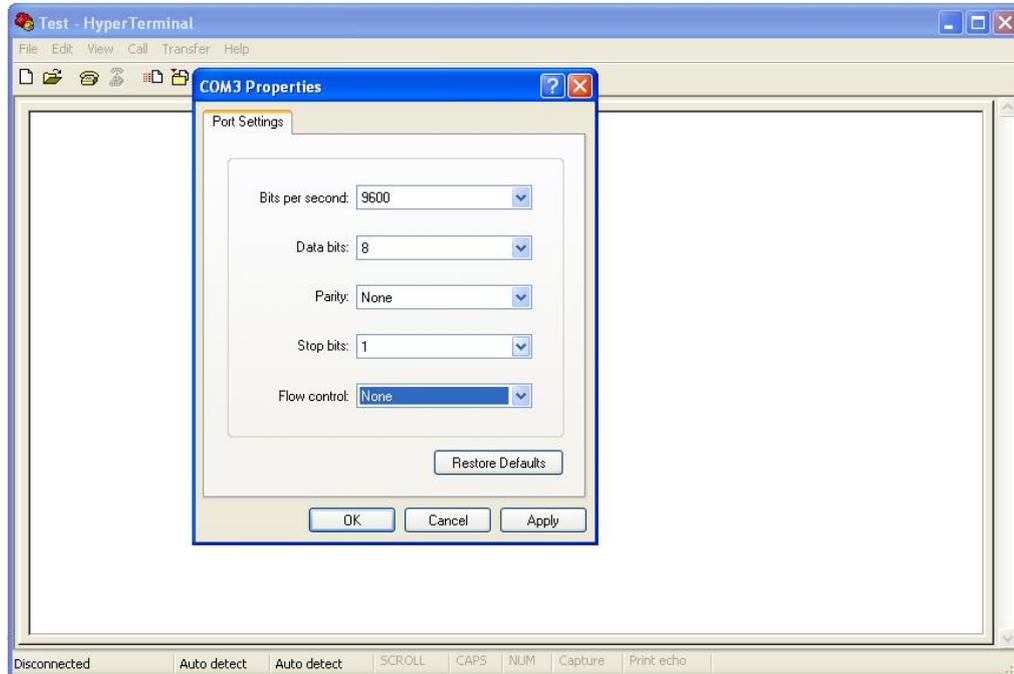


Figure 3.3-2 COM port Settings



Enter the character 'E' in the transmission blank space.

In return you will receive a string "Congratulations!" from the device ensuring the proper operation of the application loaded in the device through boot loader.

3.4 Write a file to EEPROM

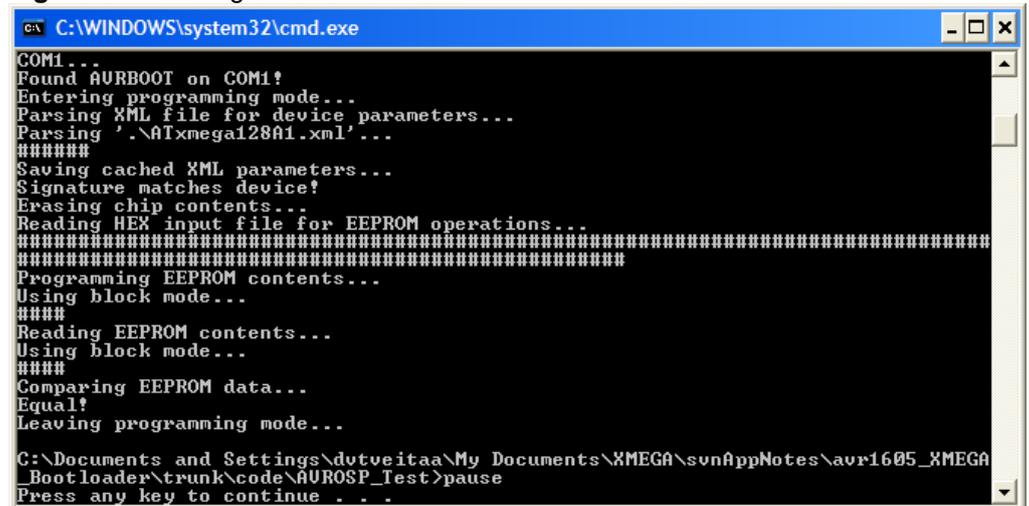
Ensure that the steps to make the device wait in bootloader section are done [Press both External RESET and SW0 buttons on STK600 and release the RESET first and then the SW0].

In the AVROSP_Test directory, run the batch file x128A1_eeeprom_write_file.bat. The file eeprom.hex containing the ASCII string "This is a test string to the XMEGA eeprom!" is written to EEPROM by using this AVROSP command:

```
AVROSP -dATxmega128A1 -e -ieeprom.hex -pe -ve
```

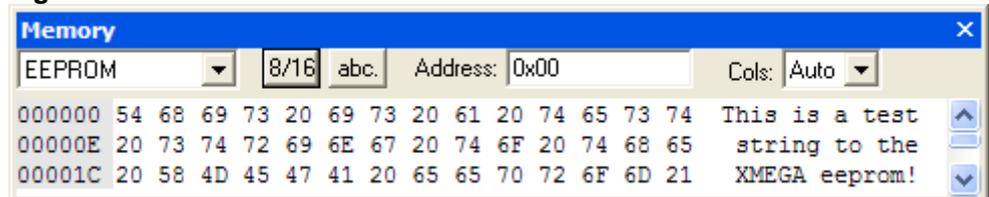


Figure 3.4-1 writing a file to EEPROM



In AVR Studio start a debug session, select memory view from the View menu. Select EEPROM and verify that the string is located at address 0x00 as in Figure 3.4-2 EEPROM visual verification to verify that the string from eeprom.hex was written correctly.

Figure 3.4-2 EEPROM visual verification



3.5 File dump from memory

3.5.1 EEPROM

Ensure that the steps to make the device wait in bootloader section are done [Press both External RESET and SW0 buttons on STK600 and release the RESET first and then the SW0].

Run the batch file x128A1_eeprom_dump.bat which has the AVROSP command given below to dump the contents of the EEPROM to the file edump.hex

```
AVROSP -dATxmega128A1 -re -oedump.hex
```

3.5.2 FLASH

Ensure that the steps to make the device wait in bootloader section are done [Press both External RESET and SW0 buttons on STK600 and release the RESET first and then the SW0].

Run the batch file x128A1_flash_dump.bat which has the AVROSP command given below to dump the contents of the FLASH to the file fdump.hex

```
AVROSP -dATxmega128A1 -rf -offdump.hex
```

4 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit <http://www.doxygen.org>. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2009 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR®, AVR Studio®, STK® and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.