APPLICATION NOTE

AVR1605: XMEGA Boot Loader Quick Start Guide

8-bit AVR Microcontrollers

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

Features

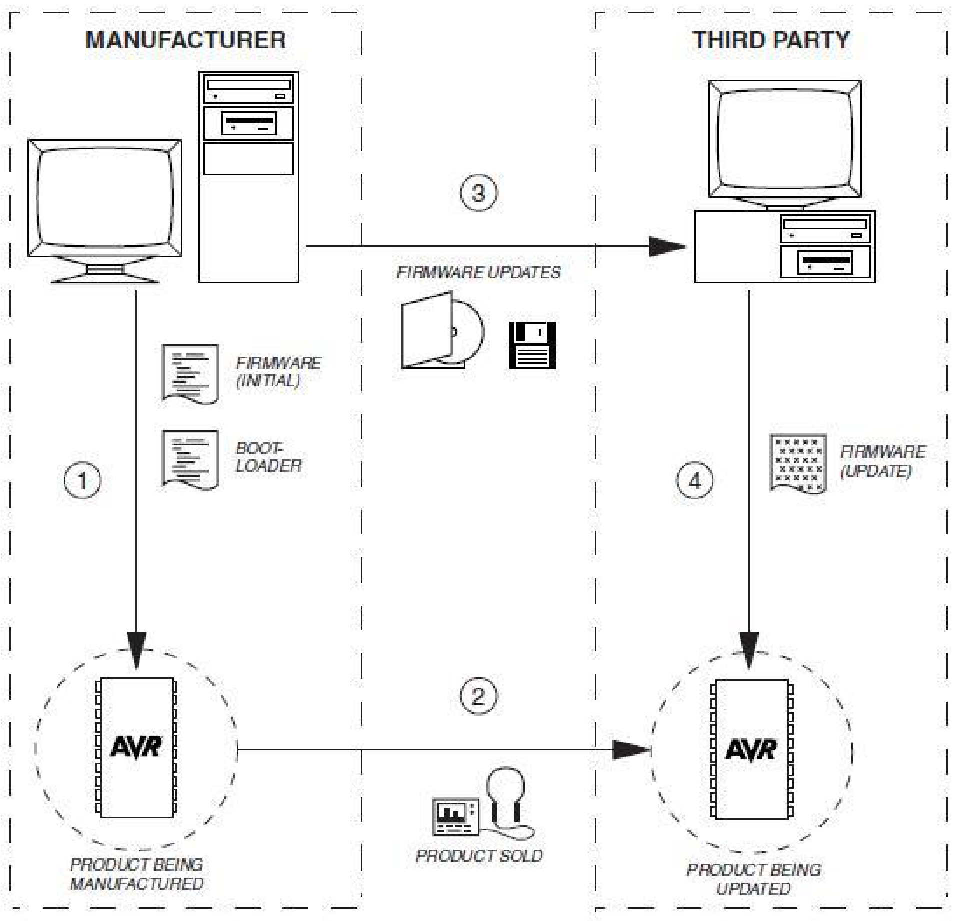
This application note describes how to use a boot loader application with one of the XMEGA family devices (i.e. ATxmega128A1) and how an AVR® with the Store Program Memory (SPM) instruction can be configured for Self-programming. The sample application communicates via the UART with a PC running the AVR Open Source Programmer (AVROSP) from the application note AVR911: AVR Open Source Programmer. 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, refer to application note AVR109: Self Programming.

Figure 1. Purpose of Boot Loader



1 Getting Up and Running

Chapter 1 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.

1.1 Hardware Setup

Section 1.1 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.

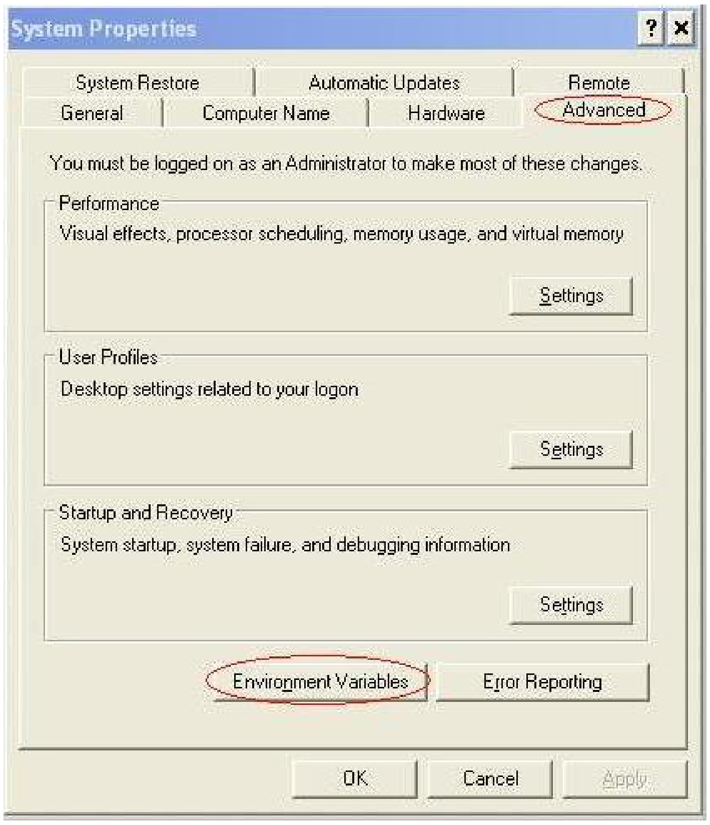
Refer STK600 User Guide, available in AVR Studio® Help to connect it with PC and to mount the device with correct routing and socket cards combination.

1.2 System Configuration

Section 1.2 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 cannot call it anywhere else. It can be set by right clicking My Computer icon and selecting the Properties option as demonstrated in Figure 1-1, Figure 1-2, and Figure 1-3. (You may need to restart the computer to enable the changes).

Figure 1-1. PATH Configuration 1



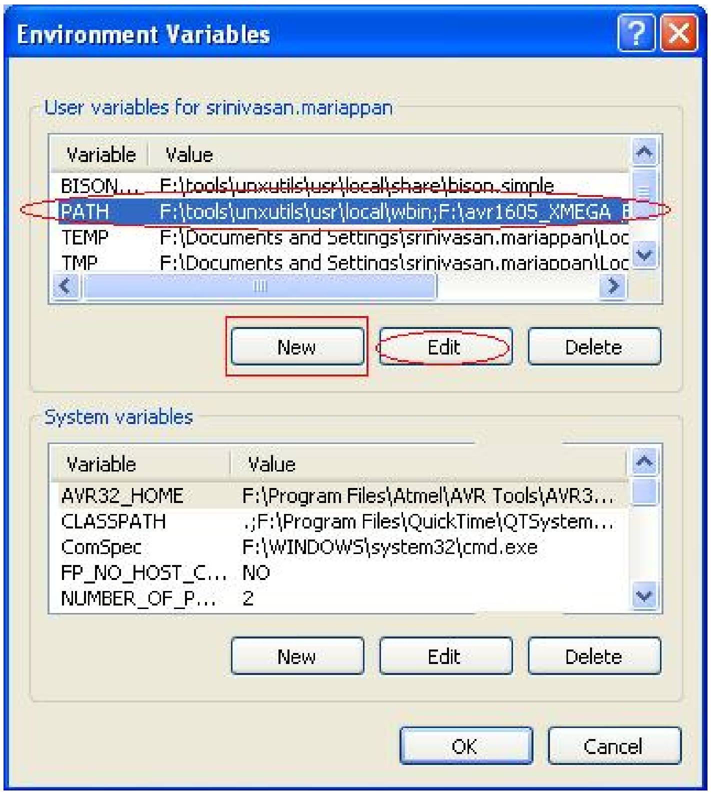
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings

User Profiles
Desktop settings related to your logon
Settings

Startup and Recovery
System startup, system failure, and debugging information
Settings

Figure 1-2. PATH Configuration 2



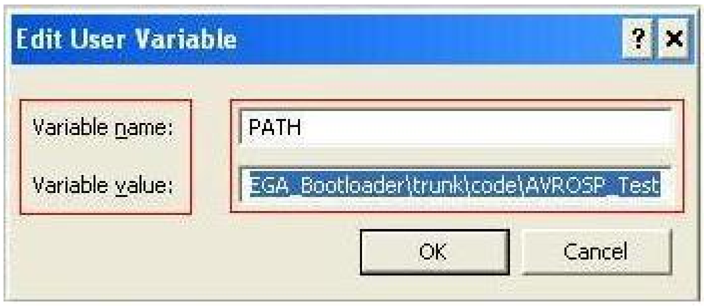
User variables for srinivasan.mariappan

Variable	Value
BISON...	F:\tools\unxutils\usr\local\share\bison.simple
PATH	F:\tools\unxutils\usr\local\wbin;F:\avr1605_XMEGA_E
TEMP	F:\Documents and Settings\srinivasan.mariappan\Loc
TMP	F:\Documents and Settings\srinivasan.mariappan\Loc

System variables

Variable	Value
AVR32_HOME	F:\Program Files\Atmel\AVR Tools\AVR3...
CLASSPATH	.;F:\Program Files\QuickTime\QTSystem...
ComSpec	F:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	2

Figure 1-3. PATH Configuration 3


Edit User Variable

Variable name: PATH

Variable value: EGA_Bootloader\trunk\code\AVROSP_Test

1.3 Starting a Debug Session

Section 1.3 gives the step by step procedure to start the debugging session on the device.

1.3.1 Starting a Debug Session in IAR

Start IARTM (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:

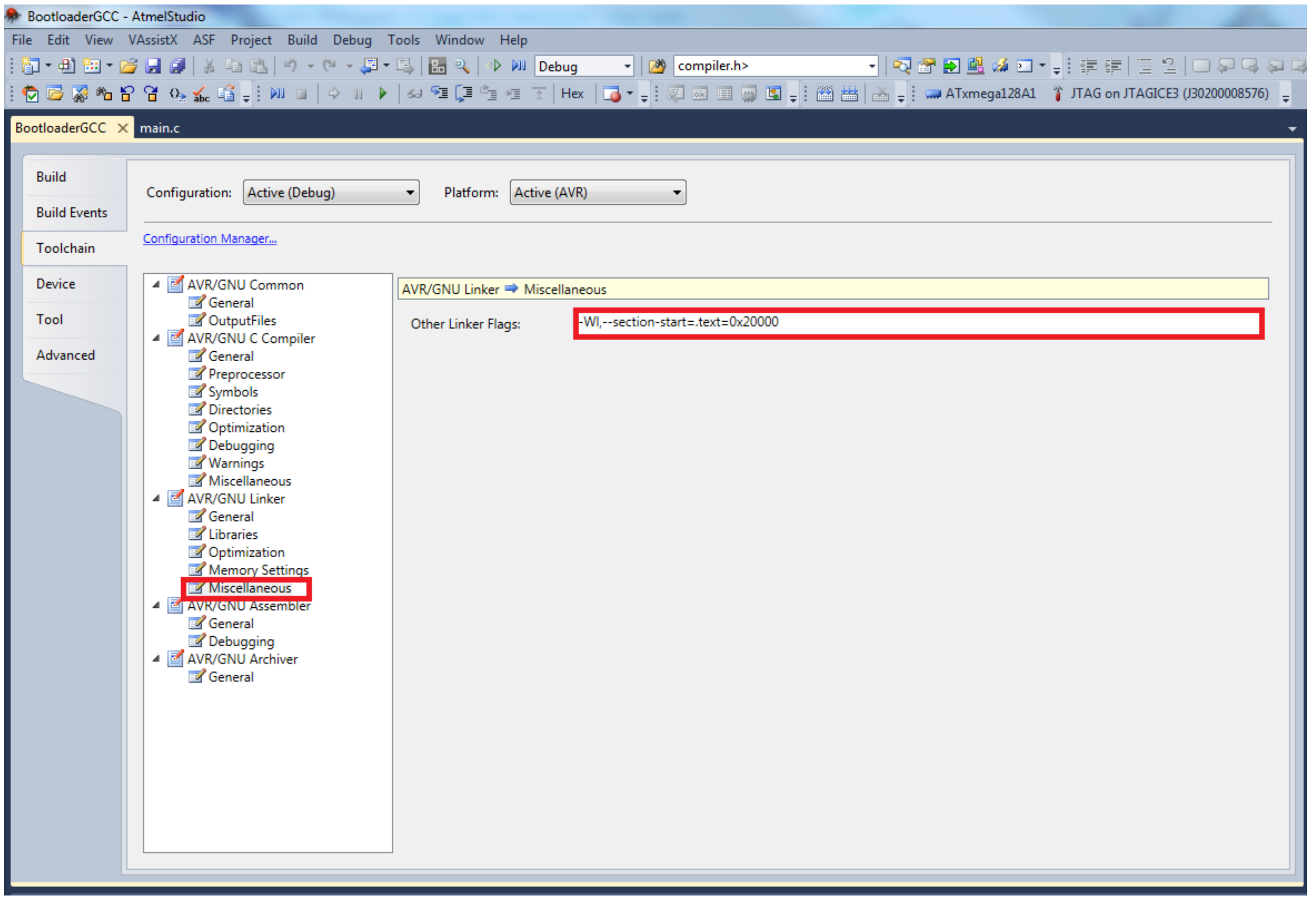
```
mode com1 Data=8 Parity=n Baud=9600 DTR=OFF RTS=OFF to=off
```

1.3.2 Starting a Debug Session in Atmel Studio

Start Atmel Studio (6.1 or later) and select the option: ”Open project/solution” Browse for the folder (BootloaderGCC) downloaded along with this Application note (hereafter called as target folder) and select BootloaderGCC.cproj for debugging. Verify that there are no errors or warnings by build and compilation.

Note: If you are using different device then the Byte address of the Boot loader must be defined in the Atmel Studio.

Project → ”project name” **properties** → **Toolchain** → **AVR/GNU Linker** → **Miscellaneous** → **Other linker Flags**: “-Wl,--section-start=.text=0x20000” (where 20000 is the Byte address of the boot loader obtained from the datasheet).



Configuration: Active (Debug)
Platform: Active (AVR)
Other Linker Flags: -Wl,--section-start=.text=0x20000

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

In Atmel 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 Atmel 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
```

2 Communication with the Boot Loader

Chapter 2 walks you through the basic steps for communicating with the boot loader through the hardware setup. The necessary setup and requirements are described along with relevant information.

2.1 Chip Erase

Section 2.1 explains how to implement the 'Chip Erase' command.

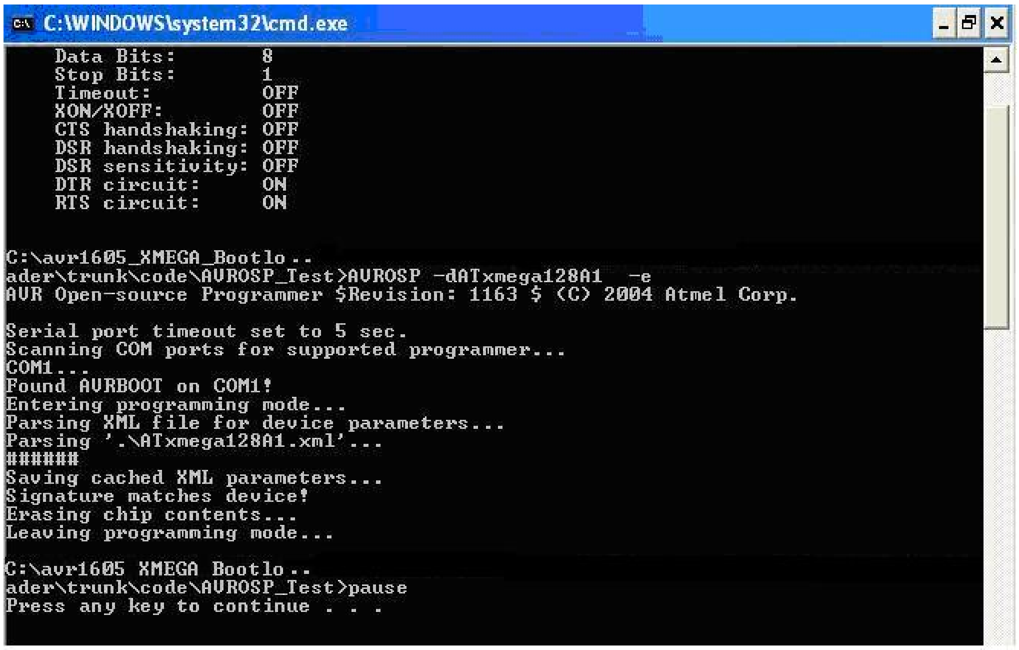
Ensure that the steps to make the device wait in boot loader section are done [Press both External RESET and SW0 on STK600 and release the RESET first and then the SW0].

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 2-1. Chip Erase



```
C:\WINDOWS\system32\cmd.exe
    Data Bits:        8
    Stop Bits:        1
    Timeout:          OFF
    XON/XOFF:         OFF
    CTS handshaking:  OFF
    DSR handshaking:  OFF
    DSR sensitivity:  OFF
    DTR circuit:      ON
    RTS circuit:      ON


C:\avr1605_XMEGA_Bootlo..
ader\trunk\code\AVROSP_Test>AVROSP -dATxmega128A1  -e
AVR Open-source Programmer $Revision: 1163 $ (C) 2004 Atmel Corp.

Serial port timeout set to 5 sec.
Scanning COM ports for supported programmer...
COM1...
Found AVRBOOT on COM1!
Entering programming mode...
Parsing XML file for device parameters...
Parsing '.\ATxmega128A1.xml'...
######
Saving cached XML parameters...
Signature matches device!
Erasing chip contents...
Leaving programming mode...

C:\avr1605 XMEGA Bootlo..
ader\trunk\code\AVROSP_Test>pause
Press any key to continue . . .
```

The command window shown in Figure 2-1, 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.

2.2 Write a File to Flash

Section 0 explains how to write/download a file to the flash.

Ensure that the steps to make the device wait in boot loader 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 -ifflash.hex -pf -vf
```

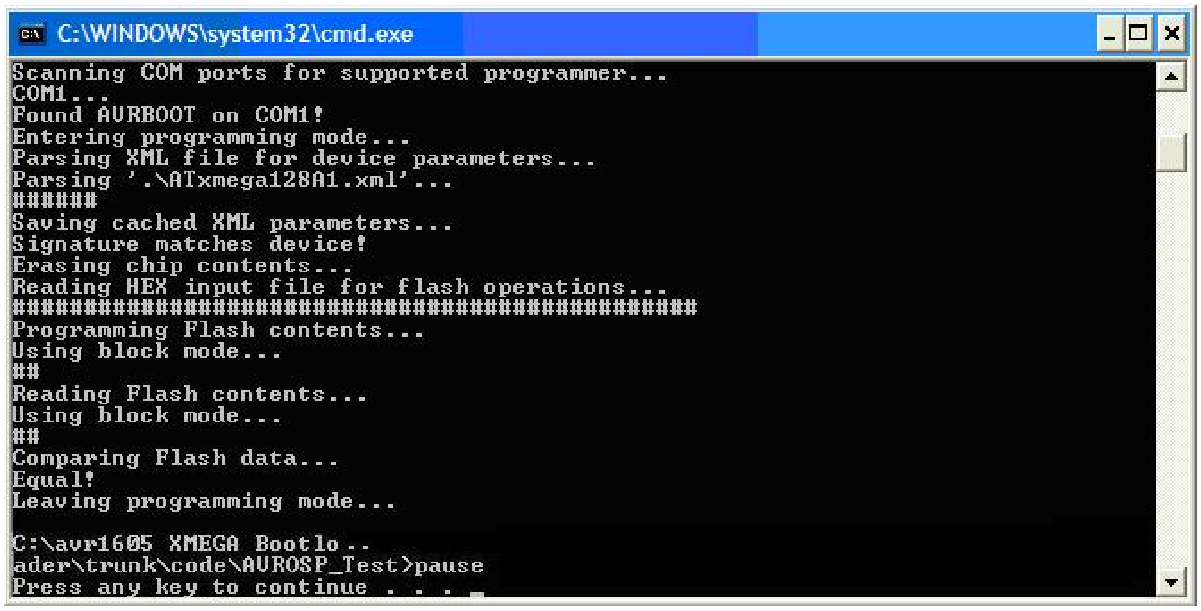
Table 2-1. Command Description

Command	Description
-e	This command will erase the flash the device.
-ifflash.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 2-2 Downloading to Flash where you can see the "Equal!" string.

Figure 2-2. Downloading to Flash



```
Scanning COM ports for supported programmer...
COM1...
Found AVRBOOT on COM1!
Entering programming mode...
Parsing XML file for device parameters...
Parsing '.\ATxmega128A1.xml'...
######
Saving cached XML parameters...
Signature matches device!
Erasing chip contents...
Reading HEX input file for flash operations...
################################################
Programming Flash contents...
Using block mode...
##
Reading Flash contents...
Using block mode...
##
Comparing Flash data...
Equal!
Leaving programming mode...

C:\avr1605 XMEGA Bootlo..
ader\trunk\code\AVROSP_Test>pause
Press any key to continue . . . _
```

2.3 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.

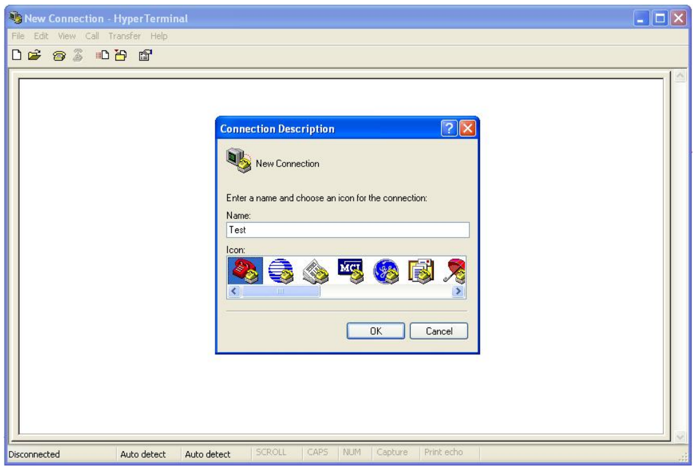
The program will jump to the application section after RESET, provided the micro switch SW0 in not pressed at the time of RESET. Alternatively, the application can be run by setting the BOOTRST fuse to Application Boot and restart the part. (Note that a start of another debugging session will perform 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 (e.g.: Test) for that session and configure the COM port settings to 9600 baud, 8bit Data, no parity, 1 stop bit and no Handshaking. Refer Figure 2-3 HyperTerminal Name and Figure 2-4 COM port Settings for more information.

Figure 2-3. HyperTerminal Name

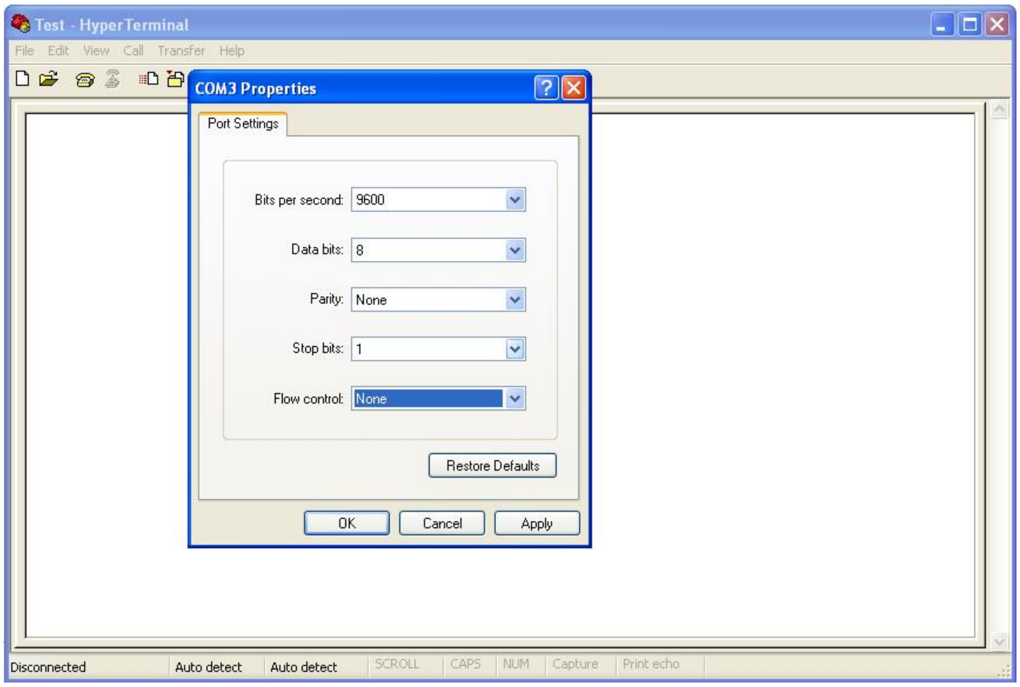


Enter a name and choose an icon for the connection:

Name: Test

Icon:

Figure 2-4. COM port Settings



Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

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.

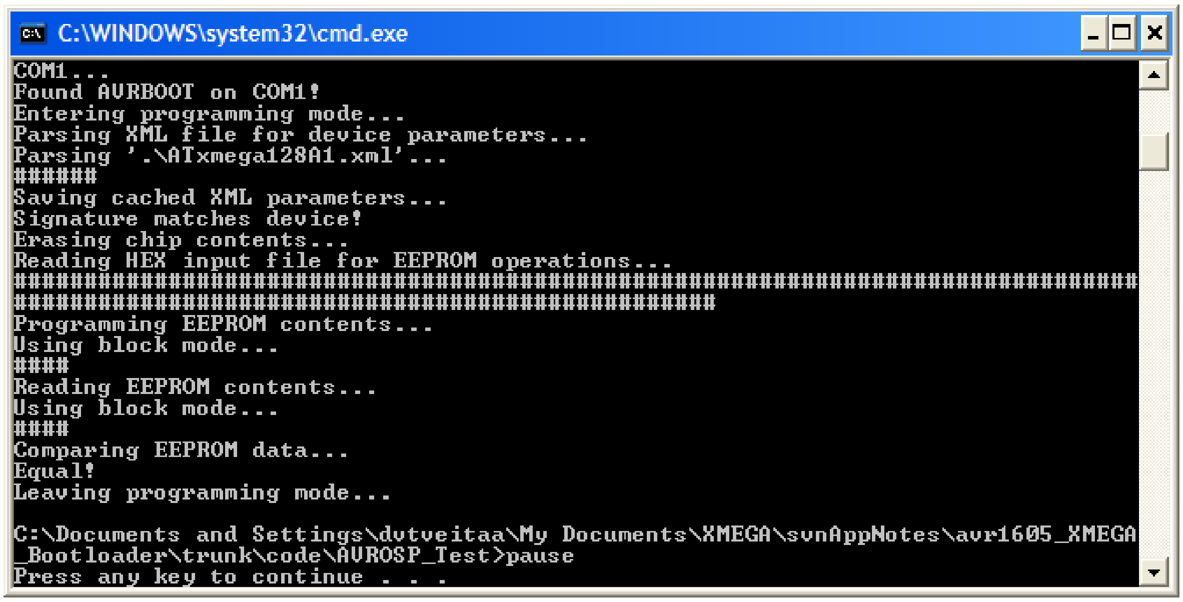
2.4 Write a File to EEPROM

Ensure that the steps to make the device wait in boot loader 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_eeprom_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 -ieeeprom.hex -pe -ve
```

Figure 2-5. Writing a File to EEPROM

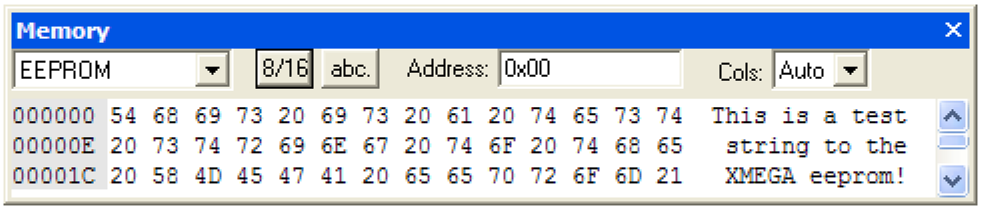


```
C:\WINDOWS\system32\cmd.exe
COM1...
Found AVRBOOT on COM1!
Entering programming mode...
Parsing XML file for device parameters...
Parsing '.\ATxmega128A1.xml'...
######
Saving cached XML parameters...
Signature matches device!
Erasing chip contents...
Reading HEX input file for EEPROM operations...
################################################################################
##################################################
Programming EEPROM contents...
Using block mode...
####
Reading EEPROM contents...
Using block mode...
####
Comparing EEPROM data...
Equal!
Leaving programming mode...

C:\Documents and Settings\dvtveitaa\My Documents\XMEGA\svnAppNotes\avr1605_XMEGA
_Bootloader\trunk\code\AVROSP_Test>pause
Press any key to continue . . .
```

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 2-6 EEPROM Visual Verification to verify that the string from eeprom.hex was written correctly.

Figure 2-6. EEPROM Visual Verification



```
000000 54 68 69 73 20 69 73 20 61 20 74 65 73 74  This is a test
00000E 20 73 74 72 69 6E 67 20 74 6F 20 74 68 65   string to the
00001C 20 58 4D 45 47 41 20 65 65 70 72 6F 6D 21   XMEGA eeprom!
```

2.5 File Dump from Memory

2.5.1 EEPROM

Ensure that the steps to make the device wait in boot loader 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 -oeedump.hex
```

2.5.2 FLASH

Ensure that the steps to make the device wait in boot loader 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
```

3 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, 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.

4 Revision History

Doc Rev.	Date	Comments
8242B	07/2014	• Added Section 1.3.1 Starting a Debug Session in IAR • Added Section 1.3.2 Starting a Debug Session in Atmel Studio • Updated to newest document template
8242A	05/2009	Initial document release.