

Atmel AVR1622: TWI Boot Loader for XMEGA



Features

- Atmel® AVR® XMEGA® boot loader
 - Uses TWI interface
- C-code for self programming
- Designed to work with Windows® utility TWIGEN (AVR1624)
- Read and write flash

1 Introduction

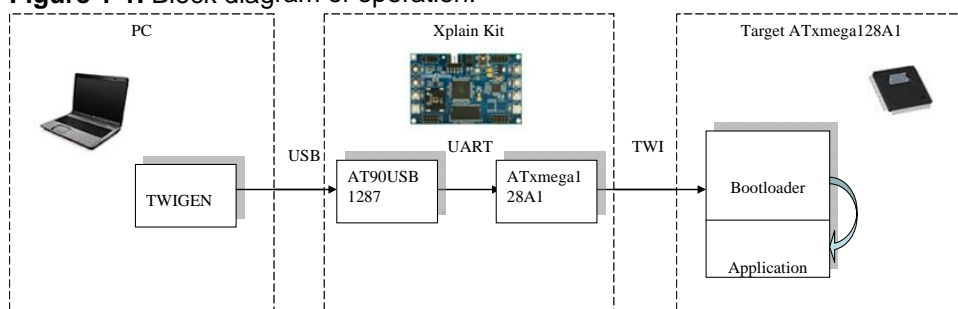
As many electronic designs evolve rapidly there is a growing need for being able to update products, which have already been shipped or sold. Microcontrollers that support boot loader facilitates updating the application flash section without the need of an external programmer, are of great use in situations where the application has to be updated on the field. The boot loader may use various interfaces like SPI, UART, TWI, Ethernet etc.

This application note describes how to use boot loader of the XMEGA family of devices to update the application section and how XMEGA can be configured for self programming. The target device used for demonstration is the Atmel ATxmega128A1 on the Atmel STK®600. It communicates via TWI interface to an Atmel AVR Xplain kit interfaced to PC through USB and running TWIGEN (from the Atmel AVR1624 application note). The Xplain kit works as a USB-to-TWI bridge, and sends the TWI packets to the target device for programming. This enables flash programming without the need for an external programmer. The example code uses an ATxmega128A1 on a STK600 target board.

The program memory for XMEGA is divided into two sections, the application program section and the boot program section. The boot loader program for updating the application flash section is placed in the boot program section. A boot loader can use any interface to download the application program, and here we use the TWI interface. Once programmed, different levels of protection can be individually applied to both the boot and application sections of the flash memory. The AVR 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](#).

Figure 1-1. Block diagram of operation.



8-bit Atmel Microcontrollers

Application Note

Rev. 8437A-AVR-09/11





2 Getting up and running

This chapter 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 explains the procedure to be followed for setting up the hardware for programming using boot loader through TWI interface.

The boot loader program given with this application note uses TWIGEN as the user interface on the PC. It implements read and write routines for reading/updating the flash section.

Since TWI cannot be used for direct communication with a PC, an Atmel AVR Xplain kit is used in between for bridge function. The Atmel AT90USB1287 microcontroller in the Xplain kit is programmed with an USB-to-UART (CDC) application and the Atmel ATxmega128A1 in the Xplain kit is programmed with an USART-to-TWI bridge application. The bridge functionality along with the TWIGEN source code is available in the Atmel application note AVR1624.

Connect PD0 and PD1 (SDA and SCL of TWI interface on PORTD) of the ATxmega128A1 on the Xplain kit to PC0 and PC1 (SDA and SCL of TWI interface on PORTC) of the ATxmega128A1 on the Atmel STK600. Connect GND from the Xplain kit to GND on the Atmel STK600. Since internal pull-ups are enabled in the application on the ATxmega128A1 of the Xplain kit, no external pull-ups are needed here.

On the STK600, connect PB0 to SW0 to enable the device to enter programming mode when switch is being pressed.

Please refer to the STK600 user guide, available in the Atmel AVR Studio® help, to mount the device with correct routing and socket cards combination and connecting port pins to switch.

2.2 Setting Xplain kit as bridge

This section explains the procedure to make the Xplain kit act as a USB-to-UART bridge between the target device ATxmega128A1 in the STK600 and PC. 'Xplain_USB.a90', 'at90usbxxx_cdc.inf' and 'XplainSerialToI2CBootLoader Bridge.hex' are available with the Atmel AVR1624.

1. Connect the Atmel AVR JTAGICE mkII to the JTAG USB on the Xplain kit.
2. Power on the JTAGICE mkII and the Xplain Kit.
3. Start the AVR Studio.
4. Open programming dialog and connect to Atmel AT90USB1287 through the JTAG interface (make sure that both JTAGICE mkII and Xplain kit are powered).
5. Select the program tab. Under 'Flash', for the input HEX file, browse to the folder containing the Xplain_USB.a90 file and program the same file.
6. Close the programming dialog.
7. A popup occurs prompting to install the software. Select 'Install from a list or specific location (Advanced)' and browse to the 'at90usbxxx_cdc.inf' file provided with the AVR1624. For more details, please refer to the AVR1624.
8. Now connect JTAGICE mkII to JTAG & PDI XMEGA connector on the Xplain kit.

9. Open the programming dialog and connect to the Atmel ATxmega128A1 through the JTAG interface (make sure that both Atmel AVR JTAGICE mkII and Xplain kit are powered).
10. Select the program tab. Under 'Flash', for the input HEX file, browse to the folder containing the `XplainSerialToI2CBootLoaderBridge.hex` file and program the same.

2.3 Programming ATxmega128A1 on STK600

This section explains how to program the ATxmega128A1 on the Atmel STK600 with the boot loader, and to program its fuses. If you want to program the hex file without debugging, please follow the steps below:

1. Connect JTAGICE mkII to the JTAG header on the STK600.
2. Power on the JTAGICE mkII and STK600.
3. Start the Atmel AVR Studio.
4. Open programming dialog and connect to ATxmega128A1 through JTAG interface (make sure that both JTAGICE mkII and STK600 kit are powered).
5. Select the program tab. Under 'Flash', for the input hex file, browse to the folder containing the `TWI_BL.a90` file and program the same file.
6. Select the fuses tab. Check the BOOTRST fuse and program.
7. Close the programming dialog.

Hence, every time the device is powered up, it enters the boot section and checks for specific condition (here – switch SW0 pressed on STK600). If the specified condition is met, it enters the self programming mode and starts programming the device with data on the TWI interface. Else, it jumps to the application section and starts executing the application code.

2.4 Starting a debug session

This section briefs the step by step procedure to be followed to start the debugging session. It is not necessary to program the flash again because the device will be programmed when debugging. IAREWB (IAR Embedded Workbench®) for AVR 5.51 was used with this.

1. Start IAREWB.
2. Select the option Open -> Workspace and browse to the folder containing the `TWI_BL.eww`.
3. Build the project and verify that there are no errors.
4. Start AVR Studio.
5. Create a new project using the `TWI_BL.dbg` file.
6. Select ATxmega128A1 as device and JTAGICE mkII as platform.
7. Open the programming dialog in AVR Studio and connect to the ATxmega128A1 through the JTAG or PDI interface (make sure that both JTAGICE mkII and STK600 are powered on).
8. Select the fuses tab and set the BOOTRST fuse and program the fuses.
9. Start a debug session in AVR Studio and run the application while keeping SW0 pressed STK600.
10. In programming mode, the program receives commands from the Atmel AVR Xplain kit which in turn passes the command received from PC through TWIGEN.

3 Communication with the boot loader

This chapter explains the basic steps for communicating with the boot loader 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 the boot loader section, are done [press both external RESET and SW0 on the Atmel STK600 and release the RESET first and then the SW0].

Double-click and run the batch program called

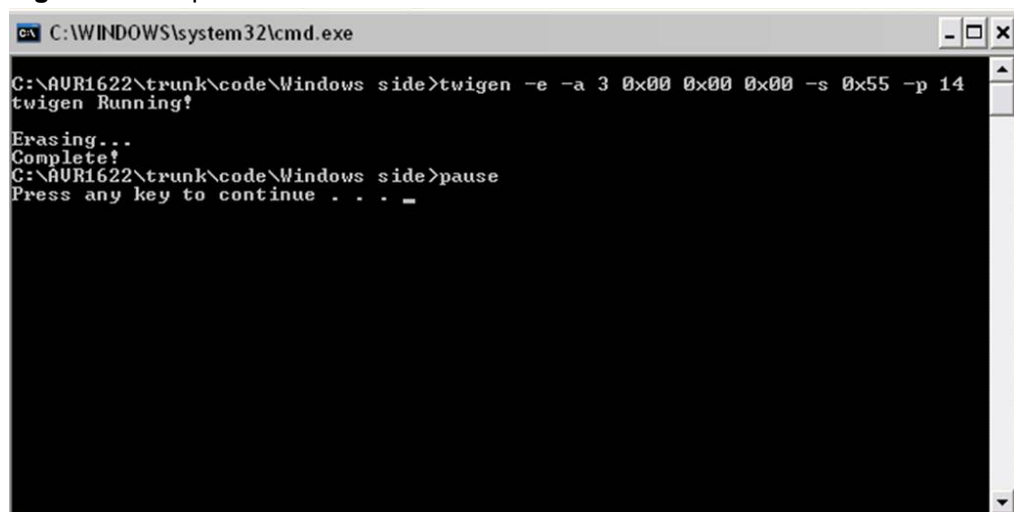
`x128A1_chip_erase.bat`

which has the device being set as Atmel ATxmega128A1, slave address as 0x55, and COM port number as 14.

The batch file runs the following command to erase the flash:

```
twigen -e -a 3 0x00 0x00 0x00 -s 0x55 -p 14
```

Figure 3-1. Chip erase.



```
C:\WINDOWS\system32\cmd.exe
C:\AUR1622\trunk\code\Windows side>twigen -e -a 3 0x00 0x00 0x00 -s 0x55 -p 14
twigen Running?
Erasing...
Complete!
C:\AUR1622\trunk\code\Windows side>pause
Press any key to continue . . . _
```

The command window shown in [Figure 3-1. Chip erase](#), should appear. The batch program performs the following operations:

1. Scans the given COM port number.
2. Enters programming mode.
3. Issues chip erase command.
4. Exits programming mode.

3.2 Write a file to flash

This section explains how to program a hex 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 the Atmel STK600 and release the RESET first and then SW0].

Double-click and run the batch program called

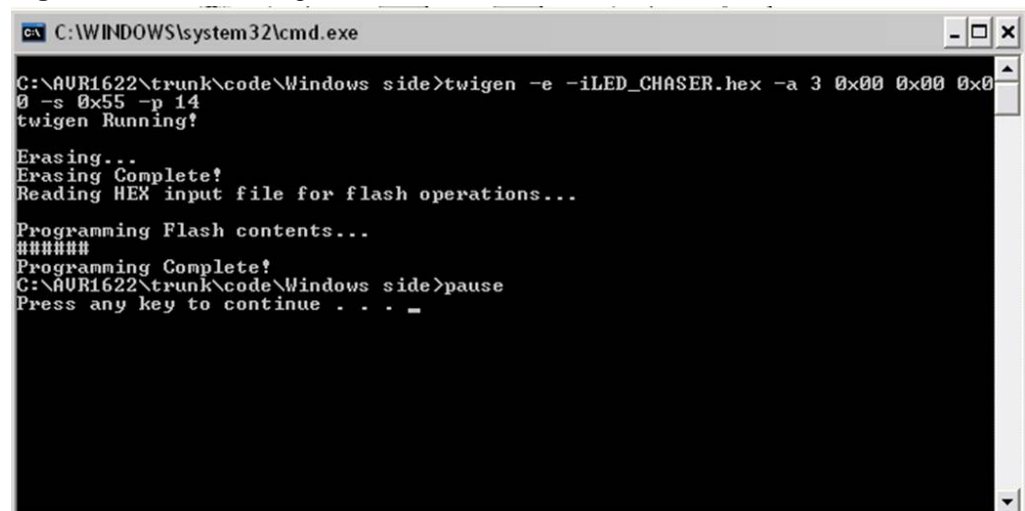
`x128A1_program_hex.bat`

which has the device set as Atmel ATxmega128A1, address from programming to done as 0x000000, slave address as 0x55 and COM port number as 14.

The batch file runs the following command to program the hex file to flash:

```
twigen -e -iLED_CHASER.hex -a 3 0x00 0x00 0x00 -s 0x55 -p 14
```

Figure 3-2. Downloading to flash.



```
C:\WINDOWS\system32\cmd.exe
C:\AUR1622\trunk\code\Windows side>twigen -e -iLED_CHASER.hex -a 3 0x00 0x00 0x00
0 -s 0x55 -p 14
twigen Running!
Erasing...
Erasing Complete!
Reading HEX input file for flash operations...
Programming Flash contents...
#####
Programming Complete!
C:\AUR1622\trunk\code\Windows side>pause
Press any key to continue . . . _
```

The command window shown in [Figure 3-2. Downloading to flash.](#) should appear. The batch program performs the following operations:

1. Scans the given COM port number.
2. Performs chip erase as explained in the previous section.
3. Enters programming mode.
4. Parses through the given hex file.
5. Programs the data in hex file to flash.
6. Exits programming mode.

3.3 Run the downloaded application

The firmware provided with this application note does a software reset after exiting programming mode.

The `LED_CHASER.hex` file runs a LED chaser on ATxmega128A1. Connect PORTD on STK600 to LED port and you can observe the output.



4 Selecting a different TWI instance

This chapter explains how to choose a different TWI instance if necessary. The sample boot loader code with this application code uses TWIC. If it is necessary to use a different TWI instance, make changes in the following locations:

1. In `flash_write_example.c`, change the interrupt vector name from `TWIC_TWIS_vect` to required `TWix_TWIS_vect` in the ISR.
2. In `twi_slave_driver.c`, change TWI instance name from TWIC to required TWIx when passing to the function `TWI_SlaveInitializeDriver`.

5 Modifying for a different device

This chapter explains how to modify the example boot loader code with this application note for a different device other than Atmel ATxmega128A1.

1. Start IAREWB.
2. Select the option Open -> Workspace and browse to the folder containing the `TWI_BL.eww` file.
3. Select the option Project -> Options.
4. In "General Options", under "Target tab", change the "processor configuration" from `--cpu=xm128a1`, ATxmega128A1 to desired device.
5. In "C/C++ compiler" under "Preprocessor" tab, change the "Defined symbols" from `__ATxmega128A1__` to the desired device.
6. In the `link_bootloader.xcl` file, change the details according to the device referring to the datasheet. Alternatively, you can also refer to the linker file for corresponding device from IAR™ for these details. IAR's linker files are available at `$TOOLKIT_DIR$\src\template\`.
7. Change `FLASH_PAGE_SIZE` in `sp_driver.s90` and `sp_driver.h` depending on the device.

NOTE

The Windows utility with Atmel AVR1624 (TWIGEN) should also be modified with corresponding page size in the project and recompiled.

6 Recommended reading

It is recommended to read the following application notes to get an overall idea about self programming and TWI interface:

- [AVR109: Self Programming](#) – This application note explains how devices with the SPM instruction can be configured for self programming. Though it is given for Atmel tinyAVR® and Atmel megaAVR® devices, it gives general information about self programming
- [AVR1316: XMEGA Self-programming](#) – This application note describes the basic functionality of Atmel AVR XMEGA self programming
- [AVR1308: Using the XMEGA TWI](#) – This application note describes how to set up and use the TWI module in the XMEGA
- AVR1624: Using ATxmega128A1 Xplain Kit as UART to TWI Bridge – This application note describes how to set up the Atmel AVR Xplain kit for making it as a UART-to-TWI bridge

7 Table of contents

Features	1
1 Introduction	1
2 Getting up and running	2
2.1 Hardware setup	2
2.2 Setting Xplain kit as bridge	2
2.3 Programming ATxmega128A1 on STK600	3
2.4 Starting a debug session	3
3 Communication with the boot loader	4
3.1 Chip erase	4
3.2 Write a file to flash	5
3.3 Run the downloaded application	5
4 Selecting a different TWI instance	6
5 Modifying for a different device	7
6 Recommended reading	8
7 Table of contents	9



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

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

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

© 2011 Atmel Corporation. All rights reserved.

Atmel®, Atmel logo and combinations thereof, AVR®, AVR Studio®, megaAVR®, STK®, tinyAVR®, XMEGA®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows® and others are registered trademarks or trademarks of Microsoft Corporation in U.S. and/or other countries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 AND 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.