

---

**Atmel AT02430: Supporting TWIGEN Interface in Serial Bootloader (AVR2054)**

---

**Atmel 8-bit Microcontroller****Features**

---

- Allows programming of .hex application images over TWI interface
- Provides support of Windows® utility TWIGEN (AVR®1624 [3])
- Supports embedded bootloader functionality required for Atmel® RF4CE stack [5]
- Supports embedded OTAU bootloader functionality for Atmel BitCloud® applications [4]

**Introduction**

---

This application note targets scenario of MCU programming with application firmware based on Atmel wireless stacks using TWI interface and Windows TWIGEN utility. Such bootloader can be used during production or for the updates in-the-field.

The package provides necessary modifications in the embedded code of SerialBootloader (AVR2054 [1]) that add support of TWI interface and protocol required by TWIGEN utility while at the same time allow reusing of bootloader functionality required by Atmel wireless stacks such as RF4CE or BitCloud.

The SerialBootloader PC tool and its serial protocol [1] cannot be used along with these modifications; instead TWIGEN utility shall be used.

This application note includes description of the procedure for bootloader source code update, compilation, hardware setup and usage with TWIGEN utility examples.

## Table of Contents

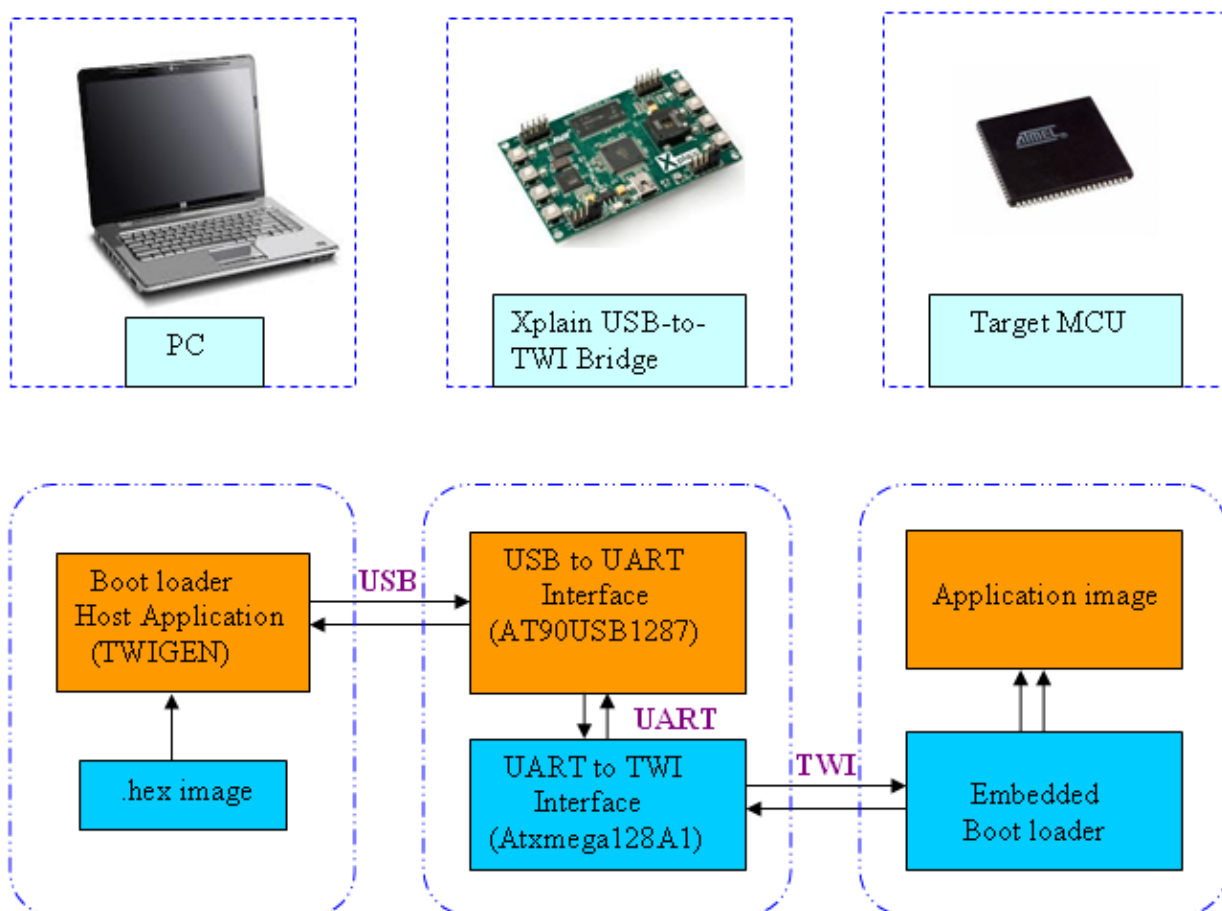
1. Overview .....	3
1.1 Supported Platforms .....	4
1.2 Getting Started .....	4
2. Programming the Boards .....	5
2.1 Programming Xplain Bridge .....	5
2.2 Programming the Target Device with Bootloader Image .....	5
3. Hardware Configuration .....	6
4. AVR TWIGEN .....	7
4.1 Steps for Programming Application Image Using TWIGEN .....	7
4.2 Steps for Xplain Kit USB Driver Installation .....	9
5. TWI Bootloader .....	13
5.1 TWI Bootloader Programming Sequence .....	13
5.2 Configuration and Compiler Options .....	15
5.2.1 Compiler Versions .....	15
5.2.2 Compilation from the Command Line and Atmel AVR Studio 4 .....	15
5.2.3 Compilation from AVR Studio 5 .....	16
5.3 OTAU/BitCloud Bootloader Features .....	16
5.4 Bootloader Configuration for RF4CE .....	16
6. References .....	17
7. Revision History .....	18

## 1. Overview

The TWI bootloader allows loading of firmware images to target devices from a PC application over the TWI interface. It is intended for use with Atmel wireless stacks, such as [IEEE® 802.15.4 MAC](#), [RF4CE](#), [BitCloud](#) and [BitCloud Profile Suite](#), but can also be used with non-wireless applications.

Figure 1-1 shows the function block of using TWI interface to load an application image on the target device. TWIGEN utility running on PC sends the firmware image of Intel® Extended format to Xplain kit through USB interface. The Atmel AT90USB1287 device on the Xplain kit receives this data over USB interface and sends it over UART interface to Atmel ATxmega128A1 device. The Atmel ATxmega128A1 on the Xplain kit receives this data over UART and passes it over TWI interface to the device running the TWI boot loader. The hex file received over TWI interface is written to flash of the target device. The TWI module on ATxmega128A1 of Xplain kit is configured in TWI master mode and the device running the TWI boot loader is configured in TWI slave mode.

Figure 1-1. Block Diagram of Using TWI Bootloader to Load an Application Image.



This application note is developed on top of the AVR2054-Serial Boot loader [1]. TWI interface is added along with the UART/USB serial communication available in the existing AVR2054 application. It is possible to select the required serial interface from the configuration file. The package that comes with this application note contains precompiled images of the embedded bootloader application for a wide set of configurations as well as its source code with projects files for different toolchains.

This package provides Atmel AVR Studio® 5.1 project files, IAR™ project files, and makefiles to build application using AVR Studio 5.1, IAR Embedded Workbench®, or the command line. The PC application software TWIGEN uses executable file called twigen.exe which is located in the \Windows side folder in the AVR1622 [2] software package.

## 1.1 Supported Platforms

Among the MCUs supported in Atmel Wireless stacks, TWI bootloader is supported on a set of Atmel microcontrollers and development boards as shown in Table 1-1. Note that some Atmel microcontrollers have more than one TWI module; based on available resource, any one of the TWI module can be used for this application. It is necessary to select the correct TWI module from the configuration file in the firmware at compile time. This application also requires an external push button to enable the device to enter the programming mode when the button is pressed. Any of the available I/O port pins configured as output can be connected to an external pushbutton/switch for this purpose. See Table 3-1 for the recommended TWI interface and GPIO connection on the supported platform.

**Table 1-1. Supported MCU's and Corresponding Boards and Modules.**

MCU (Atmel)	Supported Modules	Development Board/Kit
ATmega1281 ATmega2561	ATZB-24-B0 ATZB-24-A2 ATZB-A24-UFL ATZB-900-B0	MeshBean, RCB230/231/212 <sup>(1)</sup>
ATmega128RFA1	N/A	RCB128RFA1 ATmega128RFA1-EK1
ATxmega256A3 ATxmega256D3	N/A	Atmel STK®600, REB2xxED-EK

Note: 1. These are not Atmel products; see <http://www.dresden-elektronik.de> for purchase.

## 1.2 Getting Started

This application note is the enhancement of the AVR2054. For package content, structure and recommended fuse bit settings, the AVR2054: Serial Bootloader User Guide [1] can be referenced. The TWI bootloader is typically used in the following way:

1. Download AVR2054 and unpack the package to a folder on the PC's hard drive.
2. Unpack the package that comes along with this application note (AT02430) and copy the folders present inside AT02430.
3. Paste the copied AT02430 content inside the folder \Serial\_Bootloader\_2\_1\_0 of AVR02054\Serial\_Bootloader\_2\_1\_0.
4. Unzip AVR1622.zip and AVR1624.zip files available in the folder PC\_Bootloader\_Setup of AT02430\PC\_Bootloader\_Setup to get twigen.exe PC tool and USB-to-TWI Xplain Bridge image files.
5. Program the target device with an image (prebuilt image is available in AT02430\Embedded\_Bootloader\_images) of the TWI bootloader via JTAG.
6. Configure fuse bits for the bootloader support (see section "Fuse bits settings" in the AVR2054 [1]).
7. Program the Xplain kit with required images to communicate with TWIGEN (see Section 2.1).
8. Make proper connection between device and Xplain kit (see Chapter 3).
9. Run twigen.exe after specifying the connection settings, the application image in Intel Extended HEX format, and upload the image to the target device (see Chapter 4 for the detailed instruction).

## 2. Programming the Boards

This chapter explains the guidelines for programming firmware images on the target device and the Xplain Bridge.

### 2.1 Programming Xplain Bridge

This section explains the procedure to make the Xplain kit act as a USB-to-UART bridge between the target device and the PC. The required firmware images 'Xplain\_USB.a90', 'at90usbxxx\_cdc.inf' and 'XplainSerialToI2CBootLoaderBridge.hex' are available in the AVR1624.zip file of directory path AT02430\PC\_Bootloader\_Setup.

1. Unzip the AVR1624.zip found in the directory path AT02430\PC\_Bootloader\_Setup.
2. Connect the Atmel AVR JTAGICE mkII to the JTAG USB header on the Xplain kit.
3. Power on the JTAGICE mkII and the Xplain Kit.
4. Start the Atmel Studio.
5. Open the device programming dialog and select MCU as Atmel AT90USB1287 (make sure that both the Atmel AVR JTAGICE mkII and Xplain kit are powered).
6. Select the Memories tab. Under 'Flash', for the input HEX file, browse to the folder AVR1624\Xplain\_Bridge\_Hex containing the Xplain\_USB.a90 file and program the same file.
7. Select the Fuses tab and make fuse setting as EXTENDED: 0xF3, HIGH: 0x99 and LOW: 0x5E.
8. Close the programming dialog.
9. A popup occurs prompting to install the driver for the USB. Select 'Install from a list or specific location (Advanced)' and browse to the 'at90usbxxx\_cdc.inf' file provided with the AVR1624 in the path AVR1624\Xplain\_Bridge\_Hex. See Section 4.2 for driver installation guidelines.
10. Now, connect the AVR JTAGICE mkII to the JTAG & PDI XMEGA® connector on the Xplain kit.
11. Open the programming dialog and connect to the Atmel ATxmega128A1 through the JTAG interface (make sure that both AVR JTAGICE mkII and the Xplain kit are powered).
12. Select the Memories tab. Under 'Flash', for the input HEX file, browse to the folder AVR1624\Xplain\_Bridge\_Hex containing the XplainSerialToI2CBootLoaderBridge.hex file and program the same.
13. Select the Fuses tab and make fuse setting as FUSEBYTE0: 0xFF, FUSEBYTE1: 0x33, FUSEBYTE2: 0XF9, FUSEBYTE4: 0XFE and FUSEBYTE5: 0xFF.

### 2.2 Programming the Target Device with Bootloader Image

Programming using the TWI Bootloader requires that the embedded bootloader code is loaded to the device via JTAG. The precompiled firmware images for the embedded bootloader, which is needed to load the application image from a PC to the device, are included in this package. Images that shall be loaded to target device via JTAG may be found under the \Embedded\_Bootloader\_images\target\_device directory in the package. For example the TWI Bootloader image Bootloader\_Atmega128rfal\_TWI.hex for ATmega128RFA1 is available in \Embedded\_Bootloader\_images\Atmega128rfal.

The Atmel Studio and AVR JTAGICE mkII emulator are recommended for loading an embedded bootloader image.

**Note:** To work correctly the embedded bootloader requires specific fuse bit settings that depend on the target platform and bootloader functionality (see section "Fuse bits settings" in the Atmel AVR2054 [1]).

### 3. Hardware Configuration

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

Since TWI cannot be used for direct communication with a PC, an Atmel AVR Xplain kit is used in between for bridge functionality. 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 (see Section 2.1).

Connect PD0 and PD1 (SDA and SCL of TWI interface on PORTD) of the XMEGA PORT D header of the ATxmega128A1 on the Xplain kit to SDA and SCL of the selected target board TWI module. Table 3-1 shows the recommended TWI interface connection on the supported platform. Connect SCL of the Xplain Bridge with SCL of target board and similarly for SDA connection. The GND on the Xplain kit and the GND on the target board should be connected together if different power supplies are used for the boards. Since internal pull-ups are enabled in the application on the ATxmega128A1 of the Xplain kit, no external pull-ups are needed on the target device.

On the target board, the selected GPIO pin is to be connected to a push button to enable the device to enter programming mode when button is being pressed. If the pushbutton is not available on the target board an alternate way to put the target MCU to programming mode is to connect the GPIO pin to the GND using a jumper for one second during power ON.

The selected GPIO pin is to be connected to a LED to indicate the device is in programming mode when the button is being pressed. Table 3-1 shows the recommended GPIO connection with the available pushbutton/switch and LED on the supported platform.

**Table 3-1. Recommended TWI Interface and GPIO Connection on Supported Platforms.**

Platform	Supported TWI Interface on Target	TWI Pinouts on Supported Boards	Default TWI Interface	GPIO to Push Button/Switch Interface	GPIO to LED Interface
ATxmega256A3 REB-CBB (REB2xxED-EK)	1.TWIE (SDA-PE0 SCL-PE1)	1. Pin1 & Pin2 on PORTE header of REB-CBB Board. GND: Pin9 of PORTE header	TWIE	PB3 connected to Switch T1	PB2 connected to LED D3
ATxmega256A3 on STK600	1.TWIC (SDA-PC0 SCL-PC1)  2. TWIE (SDA-PE0 SCL-PE1)	1. Pin1 & Pin2 on PORTC header of STK600. GND: Pin9 of PORTC header 2. Pin1 & Pin2 on PORTE header of STK600 GND: Pin9 of PORTE header	TWIE	Connect PB3 of PORTB header to SW0 of SWITCHES header using jumper wire	Connect PB2 of PORTB header to LED0 of LEDS header using jumper wire
ATxmega256D3 on STK600	1.TWIC (SDA-PC0 SCL-PC1)	1. Pin1 & Pin2 on PORTC header of STK600. GND: Pin9 of PORTC header	TWIC	Connect PB3 of PORTB header to SW0 of SWITCHES header using jumper wire	Connect PB2 of PORTB header to LED0 of LEDS header using jumper wire
ATmega128RFA1 on STK600	1.TWID (SCL-PD0 SDA-PD1)	1. Pin1 & Pin2 on PORTD header of STK600. GND: Pin9 of PORTD header	TWID	Connect PE5 of PORTE header to SW0 of SWITCHES header using jumper wire	Connect PE2 of PORTE header to LED0 of LEDS using jumper wire

Platform	Supported TWI Interface on Target	TWI Pinouts on Supported Boards	Default TWI Interface	GPIO to Push Button/Switch Interface	GPIO to LED Interface
ATmega128RFA1 on RCB128RFA1	1.TWID (SCL-PD0 SDA-PD1)	1. Pin9 & Pin10 on EXT0 header of RCB GND:Pin29 on EXT0 header of RCB	TWID	PE5 connected to Switch T1	PE2 connected to LED D2
ATmega1281 on RCB230/231/212	1.TWID (SCL-PD0 SDA-PD1)	1. Pin9 & Pin10 on EXT0 header of RCB GND:Pin29 on EXT0 header of RCB	TWID	PE5 connected to Switch T1	PE2 connected to LED D1
ATmega1281 on MeshBean	1.TWID (SCL-PD0 SDA-PD1)	1. Pin9 & Pin10 on P2 header of MeshBean GND:Pin5 on P2 header of MeshBean	TWID	PE6 connected to SW1	PB5 connected to LED1

## 4. AVR TWIGEN

This chapter describes the necessary steps to start using TWIGEN. The executable file `twigen.exe` is the only file required to use TWIGEN. AVR1622.zip and AVR1624.zip files are available in the folder `AT02430\PC_Bootloader_Setup`.

The `twigen.exe` is located inside the 'Windows side' folder of AVR1622.zip file. The AVR1624.zip file contains the complete source code and the hex files to be programmed to the Atmel AVR Xplain kit.

Source firmware images shall be in the Intel Extended Hex format. Such images are created during the compilation of the application. A device must be programmed with an image of the embedded bootloader application with appropriate fuse settings, which will receive data sent by the TWIGEN through the Xplain Bridge and write it to the device's flash.

### 4.1 Steps for Programming Application Image Using TWIGEN

This section explains how to program a hex file to target flash using TWIGEN PC software through the TWI bootloader.

1. Unzip the AVR1622.zip file available in the folder `\PC_Bootloader_Setup of AT02430\PC_Bootloader_Setup`.
2. Browse to the folder `AVR1622\Windows side`, which contains the executable file `twigen.exe`.
3. Copy and paste the application image hex file to be loaded to the target in the directory where `twigen.exe` is pasted.
4. Ensure that the steps to make the device wait in the bootloader section are done (press the Push button on the target board, then power ON the board and release the button when the LED on the target board is ON. This LED is used to indicate that the target device is in the bootloader section).
5. Connect the Xplain Bridge to the PC's USB, which is already programmed with the required images by following the steps in Section 2.1.
6. Double-click and run the batch program called `x128A1_program_hex.bat` in the path `Windows side`, which has the device set as Atmel ATxmega128A1, flash start address as `0x000000`, slave address as `0x55` and COM port number as 14.

If the application image .hex file is `Terminal_Target.hex`, the COM port number where the Xplain Bridge is connected is 14, then the command to program the hex file to flash is:

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

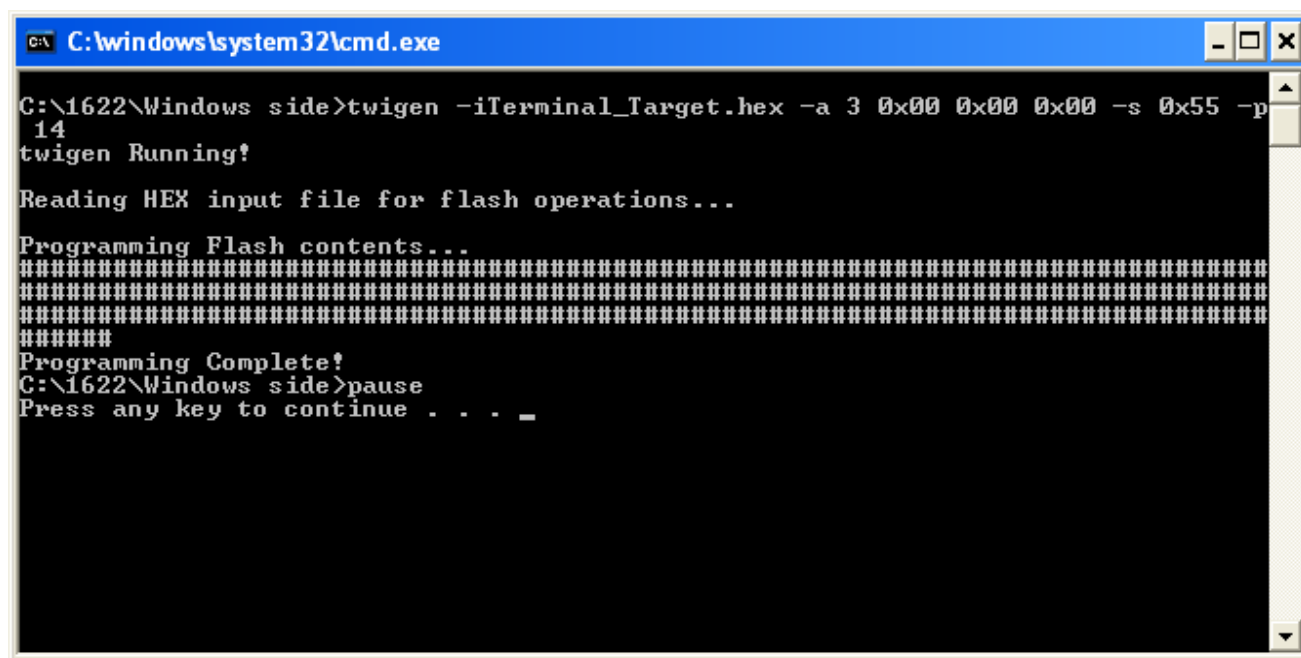
The command window shown in Figure 4-1 should appear when the batch file runs.

The batch program performs the following operations:

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

The time taken for loading the application image will be dependent on the application image size, the baud rate of TWI interface and the time required for flash erase and write. The TWI SCK baud rate is being fixed as 100kHz in the master.

**Figure 4-1. Downloading the Application Image to Flash.**



```
C:\> C:\windows\system32\cmd.exe

C:\1622\Windows side>twigen -iTerminal_Target.hex -a 3 0x00 0x00 0x00 -s 0x55 -p
14
twigen Running!

Reading HEX input file for flash operations...

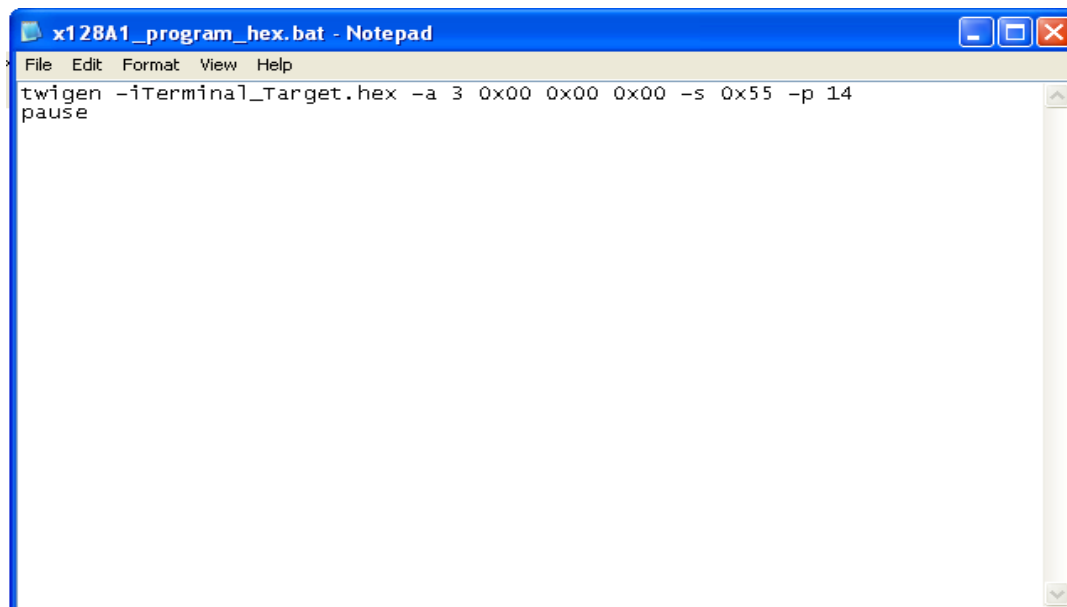
Programming Flash contents...
#####
#####
#####
#####
#####
Programming Complete!
C:\1622\Windows side>pause
Press any key to continue . . . _
```

The command can be edited for the Port number, application image file name etc., by following these steps:

1. Right click on the x128A1\_program\_hex.bat and select Edit.
2. Edit the command in Notepad as shown in [Figure 4-2](#). Refer to the Atmel AVR1624: Using ATxmega128A1 Xplain Kit as USB-to-TWI Bridge [\[3\]](#) for more details on the commands.



Figure 4-2. Editing the Command.



## 4.2 Steps for Xplain Kit USB Driver Installation

This section explains the guidelines for driver installation of the Xplain Kit USB device.

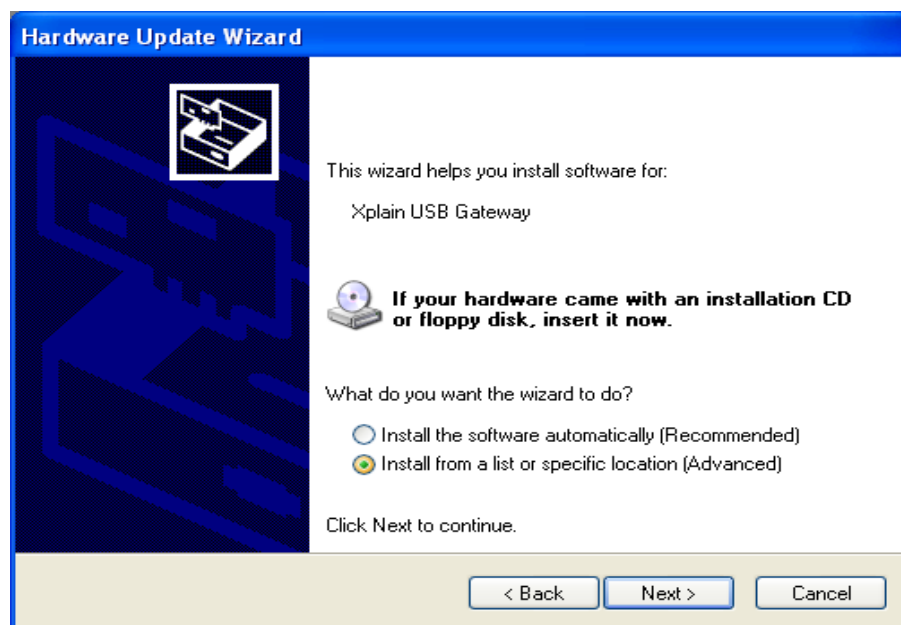
1. Connect the Xplain Kit to PC through USB. The hardware driver update wizard pop up will appear as shown in Figure 4-3. Select “Yes, this time only” in the pop-up window.

Figure 4-3. Hardware Update Wizard.



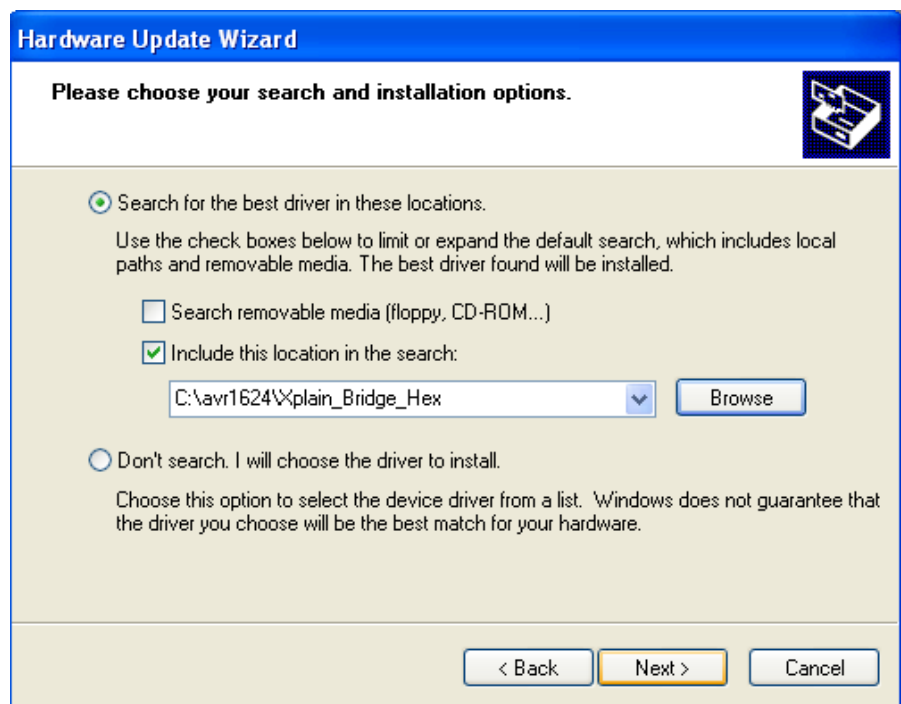
2. Select "Install from a list or specific location (Advanced)" as per [Figure 4-4](#).

**Figure 4-4. Selection for Driver Location.**



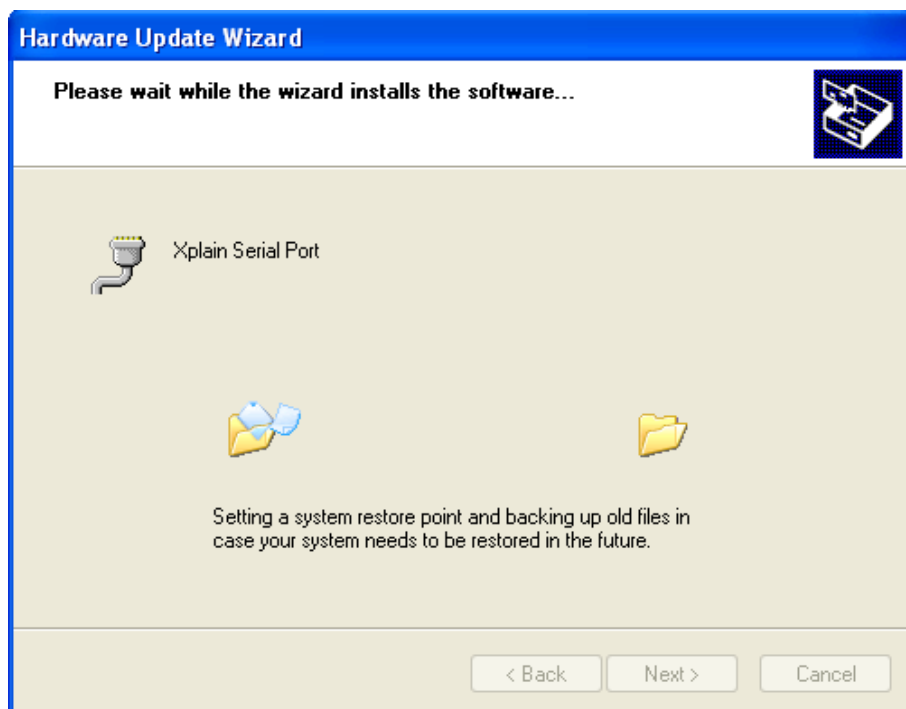
3. Browse to the path folder where `at90usbxxx_cdc.inf` is present as per [Figure 4-5](#).

**Figure 4-5. Browse to Driver Path.**

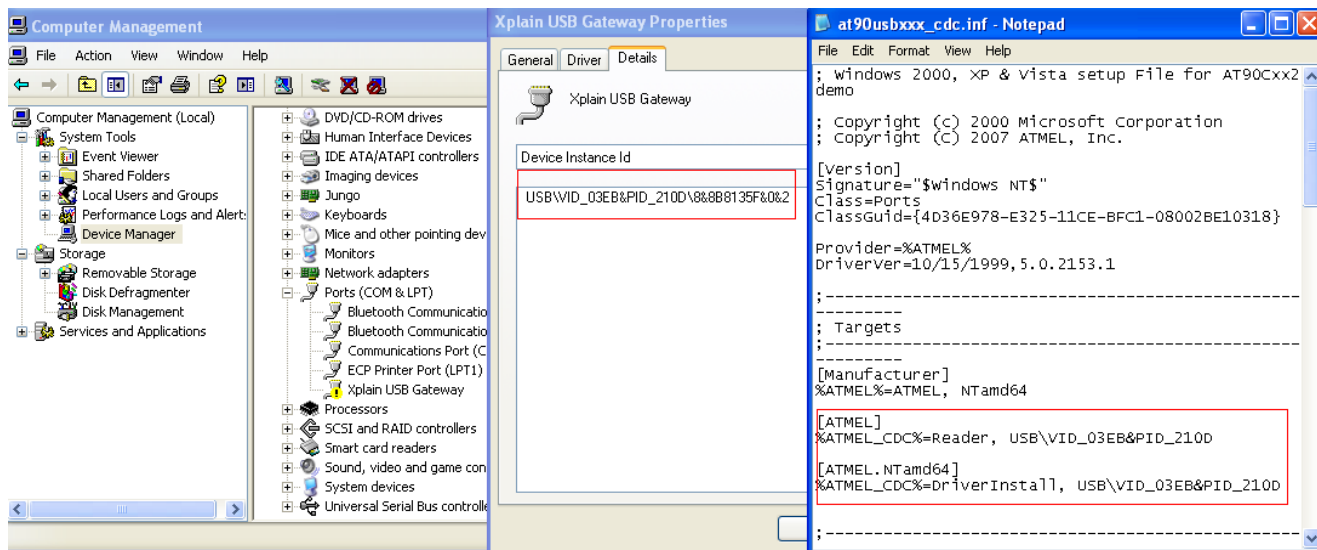


4. Driver installation will proceed as per Figure 4-6. If there is an issue with the driver installation, update the Device Instance Id in the `at90usbxxx_cdc.inf` as per connected device. Refer to Figure 4-7 for Device Instance Id.

**Figure 4-6. Driver Installation Under Progress.**

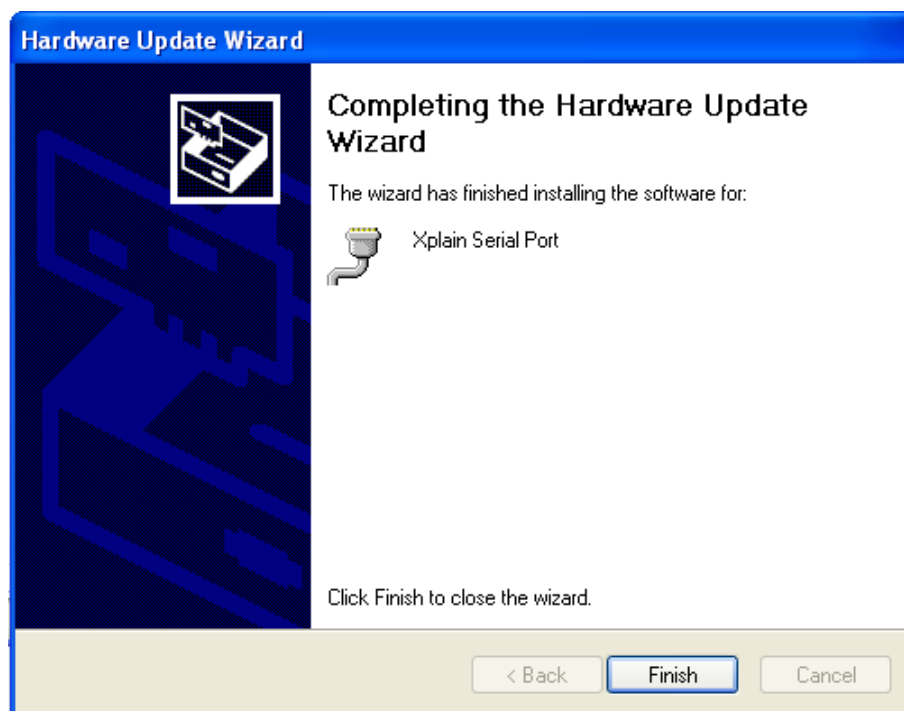


**Figure 4-7. Selection of Driver Location.**



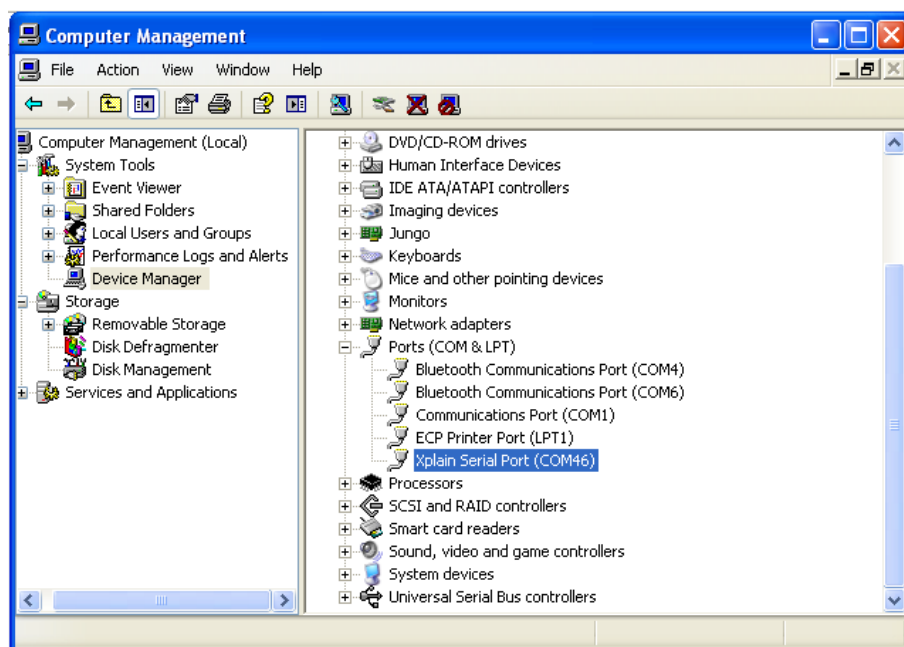
5. After the successful installation of the driver; windows will appear as per [Figure 4-8](#).

**Figure 4-8. Installation Completion.**



6. Xplain serial port appears in Device manager as shown in [Figure 4-9](#).

**Figure 4-9. Xplain Serial Port in Device Manager.**



## 5. TWI Bootloader

TWI bootloader occupies 2 to 4KB of memory and serves the purpose of loading an application image via TWI interface into the MCU's internal flash and/or EEPROM. A simple communication protocol shown in [Figure 5-2](#) is used to transfer application image to target device.

### 5.1 TWI Bootloader Programming Sequence

The TWI bootloader will process data received from Xplain kit using the following algorithm:

1. Ensure that the steps to make the device wait in boot loader section are (press the Push button on the target board, then power ON the board and release the button when the LED on the target board is ON. This LED is used to indicate the target device is in the bootloader section).
2. After the device is put in boot section by following the Step 1, the TWI bootloader waits in infinite loop, waiting for programming mode entry request from Xplain Bridge.
3. The first command from the master is making the device to enter into programming mode.
4. After the device is put in programming mode, master sends Flash erase command to erase the application section of the flash.
5. Then Hex data (Application image) is sent page by page.
6. For every valid page, the embedded bootloader responds with an ACK.
7. After the successful completion of Hex file transfer; master sends exit command to make the target to come out from the programming mode and reset the program counter.

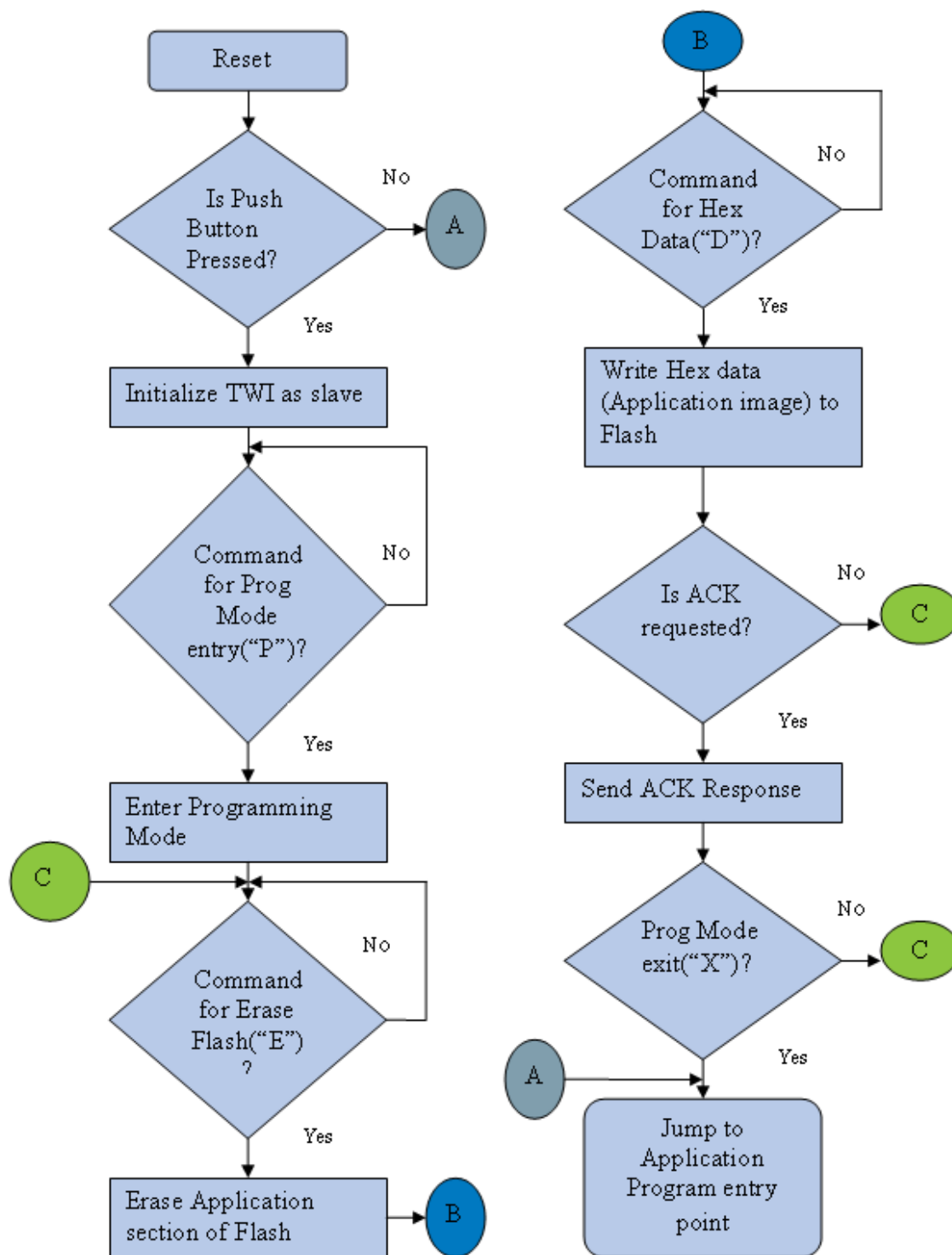
The commands used between the target device and the Xplain kit while boot loading is shown in [Figure 5-1](#).

**Figure 5-1. TWI Bootloader Communication Protocol.**

Programming Mode Entry Command (“P”-State)					
	Address bytes			Command byte	
No. of bytes of each field	LSB byte	1 byte	1 byte	1 byte	
Value	0x00	0x00	0x00	0x50	
Flash Erase Command (“E”-State)					
	Address bytes			Command byte	
No. of bytes of each field	LSB byte	SLSB byte	MSB byte	1 byte	
Value	0x00	0x00	0x00	0x45	
Hex Data Command (“D”-State)					
	Address bytes			Command byte	Application image data
No. of bytes of each field	LSB byte	1 byte	MSB byte	1 byte	256 bytes
Value	Start address of first byte of hex data			0x44	Variable
ACK Request Command					
	Address bytes				
No. of bytes of each field	LSB byte	1 byte	MSB byte		
Value	Start address of first byte of hex data				
ACK Response from Target Device					
	ACK Response				
No. of bytes of each field	1 byte				
Value	0x00				

Programming Mode Exit Command ("X"-State)				
	Address bytes			Command byte
No. of bytes of each field	LSB byte	1 byte	1 byte	1 byte
Value	0x00	0x00	0x00	0x58

Figure 5-2. TWI Bootloader Programming Algorithm.



## 5.2 Configuration and Compiler Options

Application configuration options are collected in the `configuration.h` file, which is located in the directory path `\Embedded_Bootloader_src`. Use this file to configure the following:

- Serial interface used to receive a firmware image from the host; skip to the `#ifdef` section regarding the MCU. For example, setting the TWI interface for an Atmel ATmega128RFA1 MCU is done like this:

```
#ifdef ATMEGA128RFA1
    // Use USART0
    #define USE_USART0 0
    // #define USE_USART0 1

    // Use USART1
    // #define USE_USART1 1
    #define USE_USART1 0

    // Use USB_FIFO
    #define USE_USB_FIFO 0
    // #define USE_USB_FIFO 1

    // Use TWI
    // #define USE_TWI 0
    #define USE_TWI 1
#endif
```

- With this configuration file either TWI interface or other serial interfaces like USART, USB can be selected. This means that if `USE_TWI` is being defined other interfaces cannot be used. The reason is that the protocol and the application image format used are different.
- Type of external flash device used to store the application image during an over-the-air upgrade (`TYPE_EXTERNAL_MEMORY` parameter). For example, the default setting is given by the following lines:

```
#define TYPE_EXTERNAL_MEMORY Atmel25F2048
// #define TYPE_EXTERNAL_MEMORY Atmel45DB041
```

To use an alternative option, Atmel45DB041, uncomment the corresponding line. Build configuration options for compilation with the make utility are contained in makefiles in the `makefiles` directory, while build configuration options for compilation in IAR Embedded Workbench are contained in IAR project files.

### 5.2.1 Compiler Versions

The supported IAR compilers are IAR C/C++ Compiler for AVR v6.11. The supported Windows AVR GCC (WinAVR) version is 20100110, but it is recommended to use the latest GCC compiler provided with Atmel AVR Studio 5.1.

### 5.2.2 Compilation from the Command Line and Atmel AVR Studio 4

When the bootloader application is compiled using the command line or AVR Studio 4, compilation employs Makefile placed in the `\Embedded_Bootloader_src\` folder to determine what makefile from the `makefiles` directory shall be used. In Makefile, specify `PROJECT_NAME` to choose among supported MCUs and `CONFIG_NAME` to select a particular configuration.

To open the project in Atmel AVR Studio 4, launch the `bootloader.aps` file from the `\Embedded_Bootloader_src\` folder.

### 5.2.3 Compilation from AVR Studio 5

To compile the bootloader application in Atmel AVR Studio 5.1 toolchain, open the appropriate project file from the `\Embedded_Bootloader_src\as5_projects` directory, choose a particular build configuration from the list on the toolbar, and select **Build** from the **Build** menu.

## 5.3 OTAU/BitCloud Bootloader Features

The OTAU bootloader is a version of embedded bootloader that can additionally be used to upload BitCloud application to a device through Over-the-Air upgrade (OTAU). Unlike the common bootloader, the OTAU bootloader contains a driver, which is able to transfer an image from the external flash device to the MCU's flash memory.

The OTAU bootloader is still able to write an application image received via TWI interface to the flash as a common embedded bootloader. It also supports driver for accessing external flash with OTAU applications. An application image, supporting OTAU typically includes the driver, which writes the new application image received over the air during the upgrade to an external flash device. Once a new image is loaded, the application sets a new image available status bit in EEPROM and resets. On startup after the reset the code in bootloader checks this status bit and, if it is set, transfers the image from the external flash memory to the internal MCU's flash memory.

Over-the-Air Upgrade requires a special device that will distribute the application image through the network. To start an upgrade, such device must be connected to a PC with the Bootloader PC tool installed. The Bootloader PC tool contains the *OTAU* tab (refer to [1]), which is used to initiate and control an upgrade. For further details on OTAU refer to [4].

## 5.4 Bootloader Configuration for RF4CE

The embedded bootloader configuration to support RF4CE applications is primarily the same as for the common embedded bootloader, but it also provides additional APIs to support the persistent data storage feature that can be used by RF4CE applications. The APIs provided by the bootloader and the additional APIs for RF4CE, both make use of self-programming functions. These extra APIs to support persistent data storage are written into the bootloader section. These APIs can also be used by other applications for the purposes as listed below:

- Storing the persistent data (for example, NIB) in the flash memory
- Clearing the persistent data (for example, clear NIB)
- Temporarily storing the updated image in the flash memory
- Replacing the existing application image with the updated image (swapping functionality)



## 6. References

- [1]. [Atmel AVR2054: Serial Bootloader User Guide.](#)
- [2]. [Atmel AVR1622: TWI Boot Loader for XMEGA.](#)
- [3]. [Atmel AVR1624: Using ATxmega128A1 Xplain Kit as USB-to-TWI Bridge.](#)
- [4]. [Atmel AVR2058: BitCloud OTAU User Guide.](#)
- [5]. [Atmel AVR2102: RF4Control - User Guide.](#)

## 7. Revision History

Doc. Rev.	Date	Comments
42136A	05/2013	Initial document release

**Atmel Corporation**

1600 Technology Drive  
San Jose, CA 95110  
USA

**Tel:** (+1)(408) 441-0311

**Fax:** (+1)(408) 487-2600

[www.atmel.com](http://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  
Parking 4  
D-85748 Garching b. Munich  
GERMANY

**Tel:** (+49) 89-31970-0

**Fax:** (+49) 89-3194621

**Atmel Japan G.K.**

16F Shin-Osaki Kangyo Building  
1-6-4 Osaki, Shinagawa-ku  
Tokyo 141-0032  
JAPAN

**Tel:** (+81)(3) 6417-0300

**Fax:** (+81)(3) 6417-0370

© 2013 Atmel Corporation. All rights reserved. / Rev.: 42136A-AVR-05/2013

Atmel®, Atmel logo and combinations thereof, AVR®, AVR Studio®, BitCloud®, Enabling Unlimited Possibilities®, STK®, XMEGA®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows® is a registered trademark 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 products 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.