MICROCHIP

AN2591

Getting Started with ATECC608A SecureBoot Use Case Example

Author: Kalyan C. Manukonda
Microchip Technology Inc.

INTRODUCTION

The Microchip ATECC608A device is a member of the CryptoAuthentication™ family of high-security cryptographic devices, which combine world-class hardware-based key storage with hardware cryptographic accelerators in order to implement various authentication and encryption protocols. ATECC608A provides a mechanism to support secure boot operations in a connected microcontroller unit (MCU) that can help identify situations in which fraudulent code has been installed on the host central processing unit (CPU).

CryptoAuthLib is a software-support library for the ATSHA204A, ATECC108A, ATECC508A and ATECC608A CryptoAuthentication devices, written in C. It is a portable, extensible, powerful and easy-to-use library for working with the ATSHA and ATECC family devices.

The SAM Boot Assistant (SAM-BA® application) allows In-System Programming (ISP) using a USB or UART host without any external programming interface.

This application note provides an example implementation of the ATECC608A SecureBoot feature, which operates by using the SAM-BA application. During this operation, the SAMD21 device becomes the Target, which contains the SAM-BA Monitor as the Bootloader. The Bootloader uses CryptoAuthLib to verify the user application before executing it.

PACKAGE CONTENTS

The example application contains:

- The SAM-BA Monitor application
- An example of a user application
- A Python script used to generate the Public Key and the Signature for the user application
- Updated applets and binary files for the SAM-BA software

HARDWARE AND SOFTWARE REQUIREMENTS

In order to perform the ATECC608A SecureBoot feature demonstration, the following hardware (Figure 5) and software requirements must be met.

Hardware prerequisites:

- SAMD21 Xplained Pro Evaluation Board
- CryptoAuth XPRO-B Evaluation Kit or one of the CryptoAuth-XPRO socket kits (AT88CKSCKTUDFN-XPRO or AT88CKSCKTSOIC-XPRO), connected to EXT1
- OLED1 Xplained Pro, connected to EXT3
- One Micro-B to Type-B USB interface cable

Software prerequisites:

- Atmel Studio 7
- Atmel Software Framework (ASF) 3.34
- CryptoAuthLib
- SAM-BA 2.17

Note: All pieces of software listed above can be downloaded from the Microchip website.

REFERENCES

- ATECC608A Product Details
- CryptoAuthLib
- Security ICs Overview
- SAM-BA In-system Programmer
- SAM-BA Overview and Customization Process
- Using SAM-BA for Linux on SAM Devices
- Atmel Studio 7
- SAM-BA User's Guide, which becomes available in the installation directory once the SAM-BA software has been installed.

ATECC608A APPLICATIONS

The ATECC608A device has a flexible command set, which allows usage in many applications, including the following:

- Network/IoT Node Endpoint Security Manage, node identity-authentication, session-key creation and management. Supports the entire temporary session key generation flow for multiple protocols, including TLS 1.2 (and earlier) and TLS 1.3.
- Secure Boot, which supports the MCU host by validating code digests and enabling communication keys on a “success” signal. Various configurations are available in order to offer enhanced performance.
- Small Message Encryption.
- Hardware Advanced Encryption Standard (AES) engine to encrypt and/or decrypt small messages or data such as Personally Identifiable Information (PII). The ATECC608A device directly supports the AES-ECB mode, where other AES modes are supported with the help of the host. Additionally, ATECC608A has a GFM calculation function, which supports AES-GCM.
- Key Generation for Software Download Support: local protected key generation for downloaded images. Both broadcasting of one image to other systems (each with the same decryption key), as well as point-to-point downloading of unique images per system are supported.
- Ecosystem Control and Anti-Counterfeiting validates that a system or a component is authentic and that it comes from the original equipment manufacturer.

The ATECC608A device is also compatible with the ATECC508A device when properly configured.

SECUREBOOT FEATURES

As mentioned previously, the ATECC608A provides a mechanism to support secure boot operations in a connected MCU. On power-up, the boot code within the host MCU sends the code digest and the appropriate signature to the ATECC608A device. Then, ATECC608A validates the digest by using the public key stored in the device.

If the code to be validated at boot is relatively small, then the Secure Hash Algorithm (SHA) computation engine can be used to calculate the code digest by sending the code bytes to ATECC608A.

The ATECC608A SecureBoot feature provides options for speed optimization and wire protection.

Speed Optimization

The ATECC608A SecureBoot feature includes the option to store the signature and/or the digest within the protected boundary of ATECC608A, in order to reduce the execution time. The signature and/or digest can be updated through a mode switch on the normal secure boot command, which verifies the signature and stores the signature/digest in a designated slot. Storing the signature reduces the boot time by limiting the size of the IO block that needs to be sent to the ATECC608A.

If the digest is stored, then the ATECC608A device does only a digest comparison between the host code digest in the input array and the stored digest in the designated slot. This reduces the boot time by eliminating the computation delay for the ECC verification.

Wire Protection

In some applications, it may be necessary to protect the system against an adversary who might cut the wire(s) between the ATECC608A device and the host MCU, in order to replace the results of the Verify operation with a fraudulent “success” signal. If this scenario is indicated by the mode parameter of the SecureBoot command, the input code digest can be encrypted via an XOR of the code digest, with a digest of a nonce and the IO protection secret.

Configurations and Commands

ATECC608A configuration zone controls the operation of the SecureBoot functionality of the device. In general, the SecureBoot command makes use of these configuration bits to ensure that the proper sequences are executed.

The SecureBoot feature can be configured for 3 modes of operation:

1. Full Secure Boot: both the digest and the signature are transferred to the ATECC608A device.
2. Stored Secure Boot (FullSig): the signature is stored and the digest is verified with the ECC Verify function.
3. Stored Secure Boot (FullDig): the digest is stored and will be compared without ECC verification.

The SecureBoot command enables 3 modes of operation:

1. Full: both the digest and the signature are sent to ATECC608A. The digest is verified using the signature and the public key.
2. FullStore: the digest will be sent to the ATECC608A device. The digest is verified using the signature OR the digest is stored in the device.
3. FullCopy: this command is identical to the Full command, except digest/signature is copied to the device only on successful validation.

SAM-BA MONITOR

The SAM-BA Monitor provides an easy way to program the on-chip Flash memory. The SAM-BA Monitor supports both USB and UART communications. The SAM-BA Monitor will continuously look for a start condition on the UART and/or USB interfaces.

The start condition on the USB interface is enumeration completion. When the start condition is detected, the SAM-BA Monitor enters an infinite loop and is ready for SAM-BA commands.

Start condition on the UART interface is indicated with the '#' (pound sign) character. When the SAM-BA Monitor receives this character, it awaits the SAM-BA commands.

SOFTWARE IMPLEMENTATION

The example application is provided to help demonstrate the SecureBoot feature by using the ATECC608A device. When this application is executed, it authenticates the user application using the code digest and/or the signature available on the crypto-device. It can also be used to upgrade the current user application.

Design Considerations

The following are design considerations for this implementation:

- Reserve 32 Kbytes (`0x00000000` to `0x00008000`) of Flash memory for the SAM-BA Monitor application. This is required in order to have ASF-based applications and drivers, which enables smooth integration to other family devices. It also includes full CryptoAuthLib, which enables all the features available in the library for further evaluation.
- The Bootloader application is protected using BOOTPROT and the Security Bit within the SAMD21 device, which restricts the Flash read and write access from external interference; the Bootloader area is also write protected.
- Only the UART interface is enabled to interact with SAM-BA GUI on the host. The USB-CDC interface is disabled by default.
- The SAM-BA Monitor application is not factory programmed. To use this application, it is required to follow the procedures provided in **Section "Using the SAM-BA Monitor application"**.

Figure 1 shows how the memory is split for the example application.

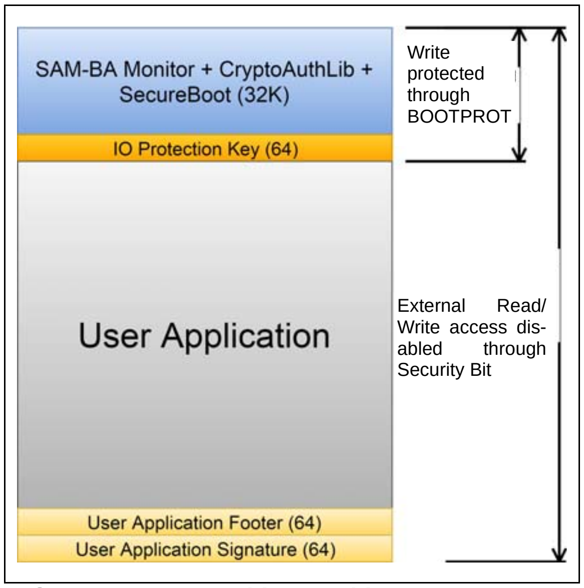

SAM-BA Monitor + CryptoAuthLib + SecureBoot (32K)
IO Protection Key (64)
User Application
User Application Footer (64)
User Application Signature (64)
Write protected through BOOTPROT
External Read/Write access disabled through Security Bit

FIGURE 1: *Memory Map.*

Secure Bootloader Flow

Figure 2 provides a high level flow chart for the SecureBoot feature implementation with the SAM-BA Monitor.

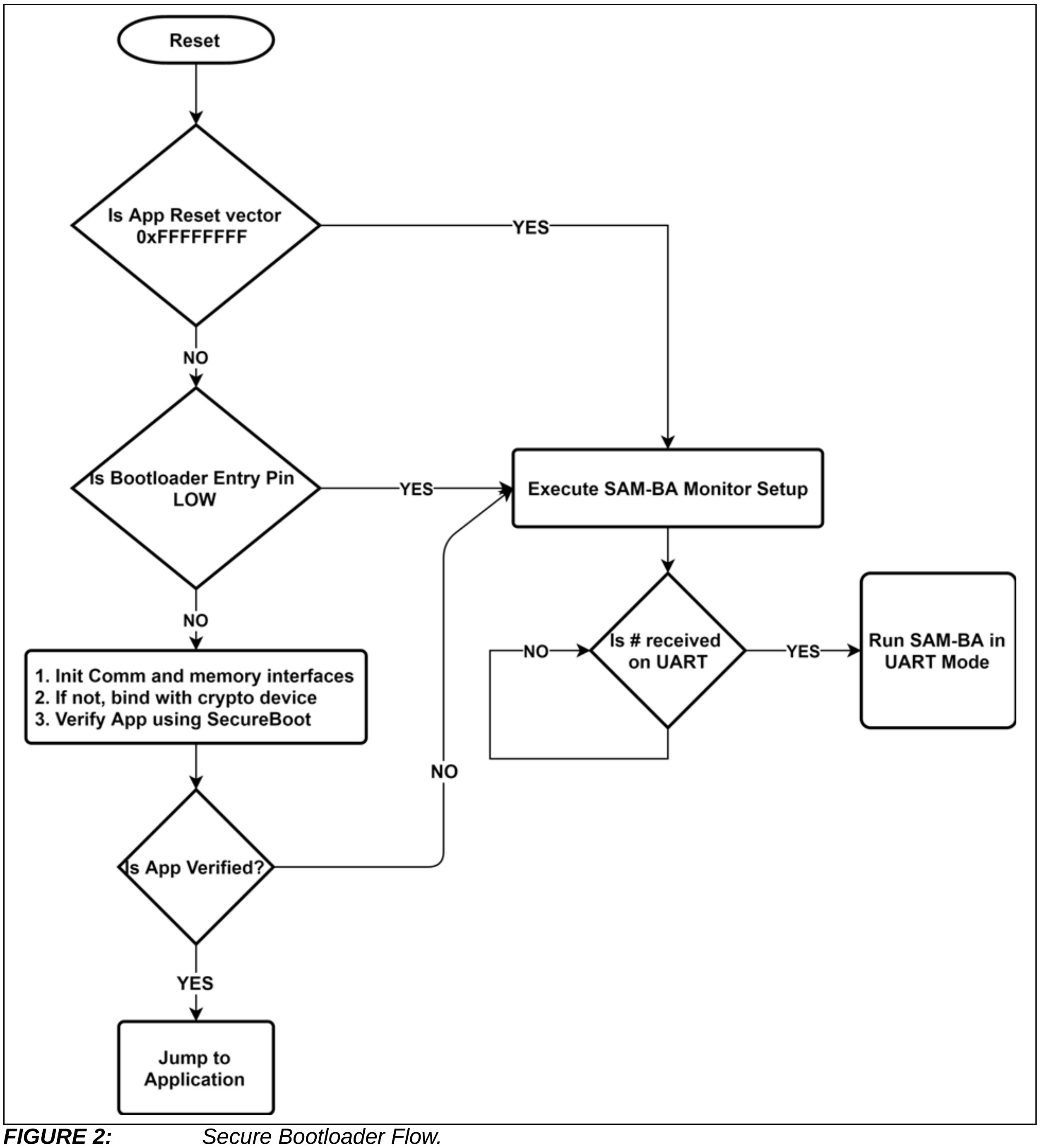

Reset
Is App Reset vector 0xFFFFFFFF
YES
NO
Is Bootloader Entry Pin LOW
YES
Execute SAM-BA Monitor Setup
NO
1. Init Comm and memory interfaces
2. If not, bind with crypto device
3. Verify App using SecureBoot
NO
Is # received on UART
YES
Run SAM-BA in UART Mode
NO
Is App Verified?
YES
Jump to Application

FIGURE 2: Secure Bootloader Flow.

Configurations

The example application can be customized by applying the conditions described in Table 1.

TABLE 1: CONFIGURATION CONDITIONS

Condition	Description
CRYPTO_DEVICE_ENABLE_SECURE_BOOT	This macro provides an option to enable or disable the SecureBoot feature.
CRYPTO_DEVICE_LOAD_CONFIG_ENABLED	This macro provides an option to enable or disable the crypto-device configuration. This requires an unlocked crypto-device attached to the Target board. When enabling this condition, the Target board loads predefined configuration data and a public key to the crypto-device. Then it locks both configuration and data zones of the crypto-device.
USER_APPLICATION_START_PAGE	This macro provides an option to adjust the user application start address. Changing this address requires modifications to be made to the user application; the SAM-BA applets adapt to reflect the new address.
IO_PROTECTION_PAGE_ADDRESS	This macro provides an option to adjust the IO protection key on the Target board.

USING THE SAM-BA MONITOR APPLICATION

The example application is downloaded to the SAMD21 MCU with the Atmel Studio software. The example application has a different bootloader size and user application start address compared to the original application from the ASF. In order to be able to use the SAM-BA GUI with this version of SAM-BA Monitor, it is required to use the files provided in the package.

SAM-BA GUI Setup

In order to use the SAM-BA GUI with the SAM-BA Monitor application, it is required to:

1. Merge the files from the SAM-BA installation directory of `[installation_directory]\applets` with the files in `Package\SAMBA_Files\applets`.
2. Merge the files from the SAM-BA installation directory of `[installation_directory]\tcl_lib` with the files in `Package\SAMBA_Files\tcl_lib`.

Activating the SAM-BA Monitor

SAM-BA Monitor (Bootloader) activation can be requested in one of the following ways:

1. **External condition:** in order to activate this condition, the user needs to pull the Bootloader entry pin low, while releasing the device from the Reset condition. A common usage is to use a push button (SW0), which can be applied as a SAM-BA Monitor trigger. The push button must be held while powering up or resetting the device.
2. **Internal condition:** this condition can be requested in the case of erased devices or when the application Reset vector (at the application's start address add 4) is blank (`0xFFFFFFFF`).

RUNNING SAM-BA

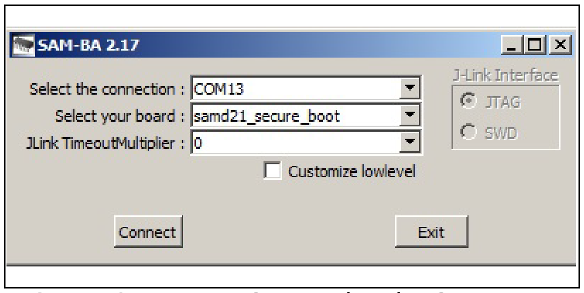
This section presents the basic steps to use the SAM-BA application on a PC running Microsoft® Windows®. For more information, please consult the SAM-BA User's Guide, which becomes available upon installation.

Connecting to the SAM-BA GUI

In order to use the SAM-BA Monitor with a UART host, connect the target board to the PC through a debug USB port.

When the SAM-BA application is executed on a Windows PC, the dialogue box in Figure 3 becomes available.

Click the **Connect** button to establish the connection with the device.



Select the connection : COM13
Select your board : samd21_secure_boot
JLink TimeoutMultiplier : 0

FIGURE 3: Connecting the SAM-BA Monitor dialogue box.

Flash Programming

After successfully connecting the device, the screen in Figure 4 is displayed.

In order to upgrade the existing application, the SAM-BA requires erasing the existing application files and downloading the newest files.

The contents of the Flash are loaded using the Flash tab (marked by “1” in Figure 4). While downloading a program to the Flash memory, the start address (marked by “2”) must match the configured value from the SAM-BA Monitor and applet (in the example application, this value is `0x08000`. Otherwise, the transfer process will be aborted.

Erase the application by selecting the “erase application area” script from the drop-down menu (3) and clicking **Execute** (4).

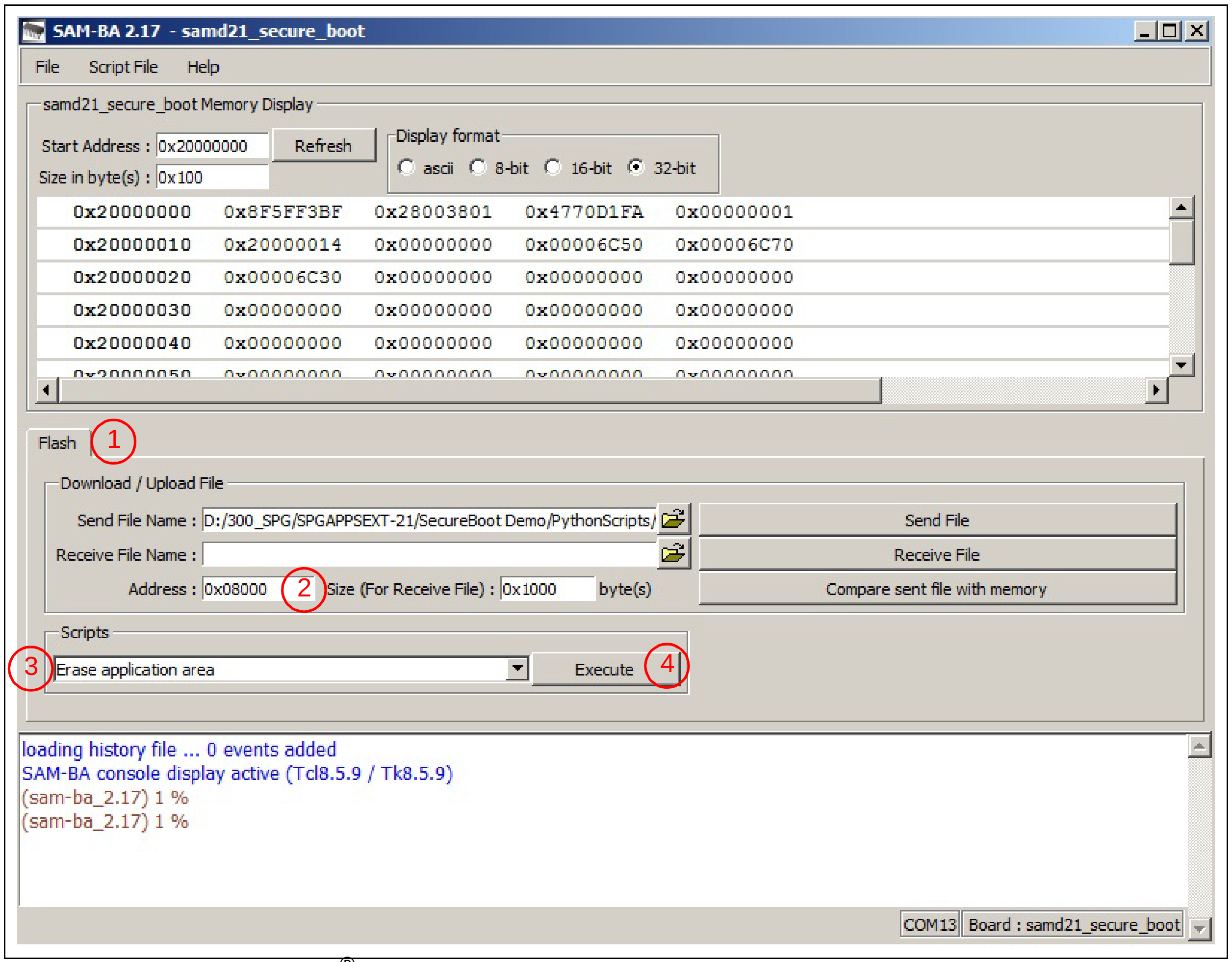
Once this is done, select the file you wish to download to the device’s Flash memory, then change the Address (2) to the `0x08000` value. Click the Send File button in order to download the firmware image to the MCU.

SCRIPTS

Table 2 shows the predefined scripts available with the SAM-BA host.

TABLE 2: PREDEFINED SCRIPTS

Script Name	Description
Erase Application Area	Erases all the application code (SAM-BA Monitor region will not be erased).
Invalidate Application	Erases the first page of the application.
Read Fuses	Returns the values of fuse settings. For details, please consult the ATECC608A Summary Data Sheet.
Read Lock Fuses	Reads the current lock settings.
Read Device ID	Reads the Device Identification register.


SAM-BA 2.17 - samd21_secure_boot
File
Script File
Help
samd21_secure_boot Memory Display
Refresh
Display format
Flash
1
Download / Upload File
Send File
Receive File
Compare sent file with memory
2
Scripts
3
Execute
4
COM13
Board : samd21_secure_boot

Start Address : 0x20000000
Size in byte(s) : 0x100
Display format: ascii 8-bit 16-bit [✓ 32-bit]

```
0x20000000  0x8F5FF3BF  0x28003801  0x4770D1FA  0x00000001
0x20000010  0x20000014  0x00000000  0x00006C50  0x00006C70
0x20000020  0x00006C30  0x00000000  0x00000000  0x00000000
0x20000030  0x00000000  0x00000000  0x00000000  0x00000000
0x20000040  0x00000000  0x00000000  0x00000000  0x00000000
0x20000050  0x00000000  0x00000000  0x00000000  0x00000000
```

Send File Name : D:/300_SPG/SPGAPPSEXT-21/SecureBoot Demo/PythonScripts/
Receive File Name :
Address : 0x08000 Size (For Receive File) : 0x1000 byte(s)
Scripts: Erase application area

```
loading history file ... 0 events added
SAM-BA console display active (Tcl8.5.9 / Tk8.5.9)
(sam-ba_2.17) 1 %
(sam-ba_2.17) 1 %
```

FIGURE 4: *SAM-BA® Window.*

RUNNING THE EXAMPLE APPLICATION

This section describes the basic steps to execute the example application. These steps are divided into the following categories:

1. Building and downloading the SAM-BA Monitor application.
2. Building and downloading the user application.
3. Provisioning the ATECC608A device using the target board.
4. Binding the target board to the ATECC608A device.
5. Verifying the SecureBoot feature.

Building and Downloading the SAM-BA Monitor Application

The SAMD21 family of devices does not include the SAM-BA Monitor as part of its ROM. Before using the SAM-BA GUI with the SAMD21 device, it is required to build and download the SAM-BA Monitor application. This can be done by using an SWD debugger.

1. Run Atmel Studio. From the top menu bar, select the **Tools** tab, then click the Device Programming menu selection.
2. From the Device Programming window, select the tool (for instance EDBG) and click **Apply**.
3. Once the EDBG tool is running, click the **Fuses** tab.
4. Set the `NVMCTRL_BOOTPROT` field to the default value of `0x07`.
5. Click the **Memories** tab and erase the chip by clicking the **Erase now** button.
6. In the Flash field, fill in the path to the Bootloader image and click the **Program** button.

Building and Downloading the User Application

A sample user application is provided in the example application package. This program is an ASF example, customized for the SAM-BA Monitor Bootloader.

The following are customizations done on the original ASF program:

1. Relocate the user application's start address to post Bootloader (for instance, starting at 32K address)
   - This example project is configured to use only 24K of user application space.
2. Create an application footer structure. The Bootloader validates the start address and the size parameters in order to match the Bootloader expectations.

```
__attribute__((section(".footer_data")))
const memory_parameters
user_application_footer =
{
USER_APPLICATION_START_ADDRESS,
(USER_APPLICATION_END_ADDRESS -
USER_APPLICATION_START_ADDRESS),
0x00010001,
{0},
};
```

3. Relocate the application footer to the last page of the image. This can be achieved through the linker script. Please refer to the `samd21j18a_flash.ld` file, provided with the example user application.
4. Compile the project to generate the file: `FREERTOS_OLED1_XPRO_EXAMPLE1.bin`
5. Protect the generated bin file by appending the signature. The example application package contains the necessary Python scripts and `key.pem` files, needed to generate signatures by using Private Key from the `key.pem` file. Follow the next steps to append the signature to the bin file:

a) Call `sboot_sign_firmware.exe` with the key file and the user application.
   - Enter Command Prompt and use the following command:
     `sboot_sign_firmware.exe -kkey.pem -bFREERTOS_OLED1_XPRO_EXAMPLE1.bin`
   - When no key file is passed to the script, `sboot_sign_firmware.exe` will generate a new key pair. This new key pair will be available in the `generated_key.pem` file.

Note: When a new key pair is used by the user, the associated Public Key should be loaded to the crypto-device using either the provision service or the target board.

b) `FREERTOS_OLED1_XPRO_EXAMPLE1.bin` will now contain the signature appended at the last location of the file.

6. Download the appended signature `FREERTOS_OLED1_XPRO_EXAMPLE1.bin` by using the SAM-BA GUI (as mentioned in **Section "Using the SAM-BA Monitor application"**).

Provisioning the ATECC608A Device Using the Target Board

The provisioning of the ATECC608A device is very important and necessary to secure the user application.

As part of this demonstration, there is an option to provision the device by using the Bootloader. The Bootloader is provided with a macro (`CRYPTO_DEVICE_LOAD_CONFIG_ENABLED`), which enables this feature. By default, this macro is disabled. When enabled, a fixed Public Key and other configurations are loaded to the crypto-device.

1. Turn off the target board and plug-in the CryptoAuth XPRO-B board to the SAMD21 Xplained Pro Evaluation Board.
   - The ATECC608A device on the CryptoAuth XPRO-B board should be in an unlocked state.
2. Turn on the target board.
3. When the macro is enabled, the target board initiates the provisioning process once it identifies an unlocked crypto-device.
   - If the crypto-device is locked, then the SAMD21 device skips the provisioning process.
4. By this point, the ATECC608A should be configured and ready to use.

Note: When a different key or different configuration setting is intended to be used, Microchip Technology Inc. offers provisioning services in order to make sure that the devices are configured with the right data through a standardized and secured provisioning process.

Binding the Target Board to the ATECC608A Device

This binding protection is meant to protect against an adversary who may cut the wire(s) between the ATECC608A device and the host MCU. It also helps to prevent the ATECC608A device from being removed from the board and used on another. This helps restrict the impact to only one board if the attacker manages to extract the IO protection secret from the MCU.

No user intervention is needed for this step. Once the target board detects a crypto-device and that the IO protection secret is not set on either the host MCU or on the crypto-device, it initiates the process.

Once the binding is done with a crypto-device, the target board initiates the BOOTPROT fuse setting. This disables all further writes from the Bootloader section of the target board.

At the end of this process, both the SAMD21 and the crypto-device are bound and have their unique IO protection secret.

Note: Once the binding between the crypto-device and the host MCU is completed, it is not possible to rebind them after erasing the IO protection secret of the host MCU.

Verifying the SecureBoot Feature

By this time, the target board is bound with the crypto-device and it is loaded with the Bootloader and user application.

Once the target board is turned on with the crypto-device, it will validate the user application and start executing it.

While the user application is executing, all the features of `FREERTOS_OLED1_XPRO_EXAMPLE` ASF project are available to evaluate.

SUMMARY OF FREERTOS_OLED1_XPRO_EXAMPLE

The `FREERTOS_OLED1_XPRO_EXAMPLE` user application is meant to demonstrate the basic usage of the FreeRTOS real-time operating system on the SAMD devices. It demonstrates the usage of tasks, queues and mutexes (semaphores).

The application is designed to run on a SAMD Xplained Pro Evaluation Board, with an OLED1 Xplained Pro Wing Board connected to EXT port.

After start-up, the application displays a pseudo-random graphic, which is continuously updated on the OLED, along with a menu bar at the bottom. The menu bar shows the selection of the user's screens. By pressing the corresponding buttons on the OLED1 Xplained Pro Wing Board, the user can select between:

- Button1 — Graph: pseudo-random graphic.
- Button 2 — Term: a text received from a terminal; for instance, the EDBG Virtual COM Port.

To add text in the terminal window, the user must connect a USB cable between the EDBG port of the SAMD MCU and a PC. The USB port of the PC will act as an EDBG Virtual COM Port. The terminal emulator software must be set for 9600 baud, 8 bits, 1 stop bit, no parity settings. The example application will echo back the received characters.

Further, the LEDs on the OLED1 Xplained Pro Wing Board are lit up for the duration of the various task loops, simply to give a visual representation of the task switching:

- LED1 represents the updating of the graphic and the handling of incoming terminal characters.
- LED2 represents the printing of text to the terminal window.
- LED3 represents the checking process of the user selection, handling display buffer and menu drawing.

Note: Note that several LEDs can be lit up at the same time if one task is waiting for another task to release a resource. In this user application, the resources are the mutexes for the display and terminal text buffer.

PORTING TO OTHER BOOTLOADERS

Converting a standard Bootloader to similar secure Bootloaders can be easily achieved by following the next steps:

1. Integrate CryptoAuthLib to the existing Bootloader, including `app\secure_boot`.
2. Integrate the `crypto_device_app.c` and the `.h` files to the project.
   - Copy and include the `crypto_device_app.c` and the `.h` files in the project.
   - Call `crypto_device_verify_app` before jumping to the user application.
   - Jump to user application only when the ATCA_SUCCESS message is displayed.
3. Setup the configurations by revisiting the configurable macros.

a) secure_boot.h.

```
#define SECURE_BOOT_CONFIGURATION              SECURE_BOOT_CONFIG_FULL_DIG
#define SECURE_BOOT_DIGEST_ENCRYPT_ENABLED     true
#define SECURE_BOOT_UPGRADE_SUPPORT            true
```

b) secure_boot_memory.h

```
#define USER_APPLICATION_START_PAGE           (APP_START_ADDRESS / NVMCTRL_PAGE_SIZE)
#define IO_PROTECTION_PAGE_ADDRESS            ((USER_APPLICATION_START_PAGE - 1) * NVMCTRL_PAGE_-
#define USER_APPLICATION_START_ADDRESS        SIZE)
#define USER_APPLICATION_END_ADDRESS          (USER_APPLICATION_START_PAGE * NVMCTRL_PAGE_SIZE)
#define USER_APPLICATION_HEADER_SIZE          (USER_APPLICATION_START_ADDRESS + (24*1024))
#define USER_APPLICATION_HEADER_ADDRESS       (2 * NVMCTRL_PAGE_SIZE)
                                              (USER_APPLICATION_END_ADDRESS - USER_APPLICATION_-
                                              HEADER_SIZE)
```

c) crypto_device_app.h

```
#define CRYPTO_DEVICE_ENABLE_SECURE_BOOT      true
#define CRYPTO_DEVICE_LOAD_CONFIG_ENABLED     false
#define IO_PROTECTION_KEY_SLOT                4
#define SECURE_BOOT_PUBLIC_KEY_SLOT           11
#define SECURE_BOOT_SIGN_DIGEST_SLOT          12
```

4. Update memory interfaces:
   - `Secure_boot_memory.c` and `io_protection_key.c` files contain access functions for the NVM/Flash. These access functions should be revisited based on the device in use.
5. Memory access restriction:
   - In this example application, the BOOTPROT fuse and the SECURITY bit are used to avoid unwanted access to the Flash memory. The user needs to implement similar restrictions based on the device in use.
6. User application updates:
   - It is required to adjust the user application to a new location as set in the Bootloader, along with a footer data, so that the Bootloader can verify its authenticity and then jump to its execution. **Section "Building and Downloading the User Application"** provides more details.

7. As a last step, the user needs to revisit the Python script provided in order to make sure that it matches the memory map, generates keys appropriately and appends the signature at the right location.

As these steps depend on the device and the Bootloader architecture, the user needs to consider other changes based on the applications.

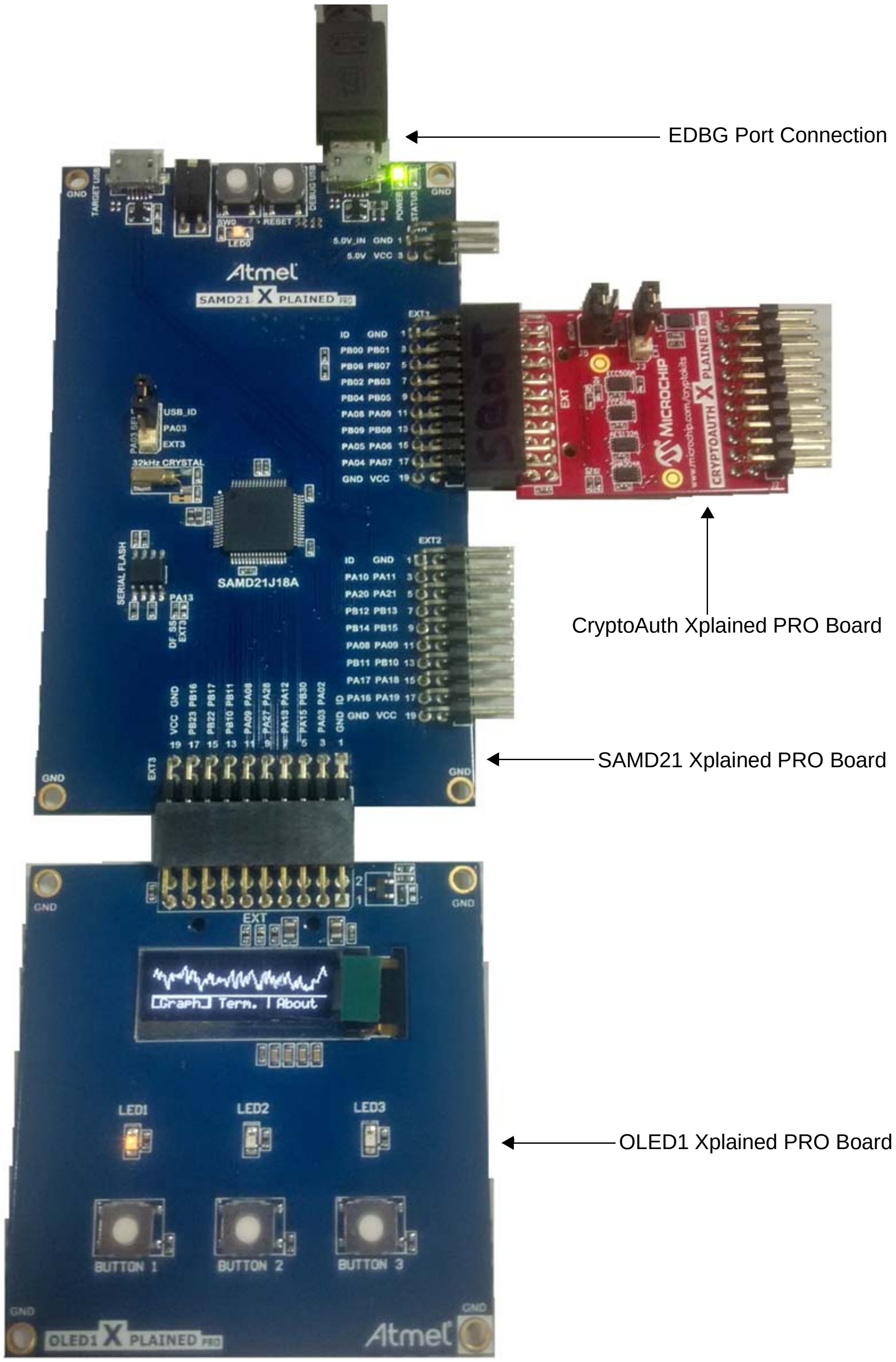


FIGURE 5: *Top View of the Example Application Setup.*

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

═ ISO/TS 16949 ═

ISBN:

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

10/25/17