
AT15004:Using SAM-BA for Linux on SAM Devices

APPLICATION NOTE

Introduction

A few of the SAM devices are pre-loaded with the Atmel® SAM Boot Assistant (Atmel SAM-BA®). This bootloader allows In-System Programming (ISP) from USB host without using any other external programming interface.

SAM-BA has been validated on various Linux distributions such as Ubuntu, Debian, Fedora, openSUSE, and Mint. However, it will work on other distributions as well. Tested distributions are listed at www.at91.com/linux4sam.

This application note describes the steps to install and use SAM-BA on some popular Linux distributions to program Atmel ARM® based devices.

Features

- Allows to program, verify, and secure an Atmel SAM device without using a device programmer
- Uses USB CDC class for communicating to the host
- Use SAM-BA Command Line mode to program SAM devices
- Source code for SAM-BA applet is available and can be customized to user's needs such as Encrypted loader and custom protocol

Table of Contents

Introduction.....	1
Features.....	1
1. Introduction to SAM-BA.....	3
2. Installing SAM-BA on Different Linux Distributions	4
2.1. Setup.....	4
2.2. Installing SAM-BA.....	4
3. Using SAM-BA	6
3.1. Understanding SAM-BA Architecture.....	6
3.1.1. SAM-BA Monitor.....	6
3.1.2. Applet Workflow.....	7
3.2. Connect device.....	8
3.3. Understanding SAM-BA GUI.....	9
3.3.1. Memory Display Area.....	10
3.3.1.1. Read Memory.....	10
3.3.1.2. Edit Memory Content.....	11
3.3.2. Memory Download Area.....	11
3.3.2.1. Flashing Firmware.....	11
3.3.2.2. Reading From Memory.....	11
3.3.2.3. Compare Memory with a File.....	12
3.3.3. Script File Functionality.....	12
3.3.3.1. Start/Stop/Reset Recording.....	12
3.3.3.2. Edit the Script File.....	12
3.3.3.3. Execute the Script File.....	12
3.4. SAM-BA in command line.....	13
4. Tested Devices.....	14
5. FAQ.....	15
6. References.....	16
7. Revision history.....	17

1. Introduction to SAM-BA

The SAM-BA GUI tool provides a means to easily program various Atmel ARM[®] processor based microcontrollers.

The key features of SAM-BA GUI are:

- performs In-System Programming through JTAG, RS232, or USB interfaces
- on-chip and on-board memories programming solutions
- can be used via a Graphical User Interface or started in batch mode from a command line
- memories and peripheral register display content
- user scripts executable from SAM-BA GUI or a shell
- operates on both Windows or Linux OS

Note:

1. SAM-BA for Linux supports only the USB interface.
2. Linux Mint 17.2 LTS and Fedora 23 will be used as the Linux distribution reference for the application note.

2. Installing SAM-BA on Different Linux Distributions

2.1. Setup

For Linux distributions, user must be a member of the dialout group to access serial devices such as RS232 serial port, USB serial, and serial modem. As SAM-BA uses the USB serial class driver to connect to the SAM devices, it is important to enable access to the serial devices for current user.

To add current user to dialout group, execute the following commands.

```
cd ~
sudo adduser linux dialout
cat /etc/group | grep dialout
```

The command line displays the following message:

```
dialout:x:20:linux
```

Note: The user name `linux` is a placeholder variable. Use appropriate value for the username.

To ensure that the changes are effective and the serial port is accessible, log out from system and log in again.

1. Download the SAM_BA for linux from Atmel website.
2. Open a terminal window and type `cd ~/Downloads`.
3. Extract the files into the home directory by executing the following command

```
#unzip sam-ba_2.15.zip -d ~/
```

This creates a `sam-ba_cdc_linux` folder in the home directory.

2.2. Installing SAM-BA

SAM-BA is a 32-bit program. It requires 32-bit libraries to be installed on 64-bit version of the operating system. To verify if the system is 32-bit or 64-bit, open the terminal and type:

```
uname --machine
x86_64 -> "64" means you are using a 64-bit distribution
```

For Linux Mint, the 32-bit libraries can be downloaded using apt-get manager. To install the 32-bit libraries, execute the following command in the terminal:

```
sudo apt-get install ia32-libs
```

For Linux Fedora, the 32-bit libraries can be downloaded using dnf or yum manager. To install the 32-bit libraries, execute the following command in the terminal:

```
sudo yum-deprecated install glibc libstdc++ linX11
sudo yum-deprecated install glibc.i686 libstdc++.i686 linX11.i686 libXScrnSaver
```

To include SAM-BA directory to the PATH variable, edit the `.bashrc` file

```
gedit ~/.bashrc
```

At the end of the file, add the following line:

```
export PATH=$PATH:$HOME/sam-ba_cdc_linux
```

Source `.bashrc` file to update the PATH variable:

```
source ~/.bashrc
```

SAM-BA for Linux has been successfully installed.

3. Using SAM-BA

3.1. Understanding SAM-BA Architecture

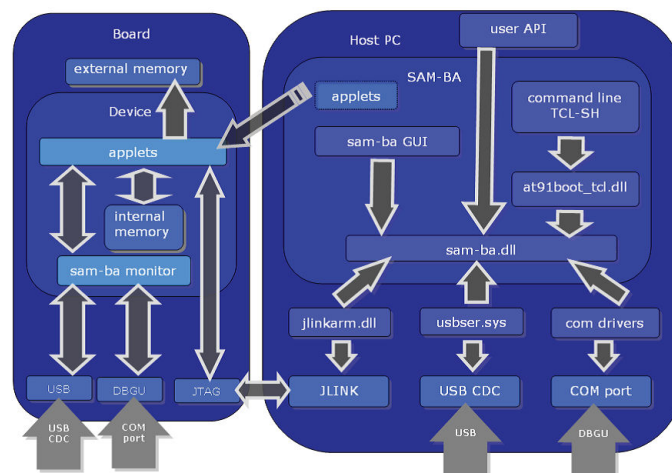
SAM-BA provides tools for programming all Atmel ARM-based microcontrollers devices . They are based on a common dynamic linked library (DLL), `sam-ba.dll`. It is an OLE COM component distributed under a DLL allowing automation tool.

It is also possible to execute the SAM-BA functions in command line using a TCL shell. An intermediate DLL (`AT91BOOT_TCL.DLL`) is used to transform TCL commands to the `sam-ba.dll`. Several communication links are available such as USB, serial port (RS232) or JTAG.

The SAM-BA has two parts, the host and the target device. The host runs on computer. It sends programming files and programming instructions over a download cable to the target. The target part is a hardware design, running in the Atmel ARM based devices. It accepts the programming data—content and required information about the target external memory device – sent by the host, and follows the instructions to write/read data to/from the external memory device.

Note: For list of supported devices, refer to the release notes of SAM-BA.

Figure 3-1. SAM-BA Architecture



3.1.1. SAM-BA Monitor

The SAM-BA monitor is a default boot program which provides an easy way to program in-situ the on-chip Flash memory. The SAM-BA Boot Assistant supports serial communication via the USB. The SAM-BA Boot provides an interface with SAM-BA Graphic User Interface (GUI). In most of SAM devices, SAM-BA monitor will be stored in ROM. So in order to execute SAM-BA monitor user needs to boot from ROM. The boot location is decided by General-Purpose NVM (GPNVM1) bit.

The GPNVM1 bit can be cleared or set through the commands “Clear General-purpose NVM Bit” and “Set General-purpose NVM Bit” of the EEFC User Interface respectively. Detailed information on SAM-BA boot strategy can be obtained from SAM-BA boot section of the device specific datasheet.

The SAM-BA Boot Program integrates an array of programs permitting download and/or upload into the different memories of the product. The SAM-BA monitor will continuously look for a successful USB enumeration. When the communication interface is identified, the monitor enters into an infinite loop waiting for various commands.

Note: The SAM devices are available with factory loaded SAM-BA monitor in the ROM. However, some of the devices such as SAM D20 or SAM 4L do not have ROM. Such devices can still utilize SAM-BA by reserving a section of Flash for SAM-BA. For more information on SAM-BA boot strategy for such devices, refer to device specific application notes for SAM-BA. Link for some of these application notes are available in [References](#) section .

3.1.2. Applet Workflow

The target handles the programming algorithm by running applets. The target switches between two modes: SAM-BA Monitor Mode and Applet Mode. The SAM-BA monitor mode is the command interpreter that runs in the ROM memory when users connect the chip with USB or COM port to the computer. It allows the computer to send or receive data to/from the target. All transfers between host and device are performed when the device is in SAM-BA monitor mode.

Under Applet Mode, the device performs programming operations and can not communicate with the host. An applet is a small piece of software running on the target. It is loaded in the device memory while the device is in SAM-BA monitor mode using TCL_Write command. The device switches from SAM-BA monitor mode to Applet mode using the TCL_Go command. The device executes the applet code. At the end of the current operation, the device switches back to SAM-BA monitor mode. An applet can execute different programming or initialization commands. Before switching to Applet mode, the host prepares command and arguments data required by the applet in a mailbox mapped in the device memory. During its execution, the applet decodes the commands and arguments prepared by the host and execute the corresponding function. The applet returns state, status and result values in the mailbox area. Usually, applets include INIT, buffer read, buffer write functions. To program large files, the whole programming operation is split by the host into payloads. Each payload is sent to a device memory buffer using SAM-BA monitor command TCL_Write. The host prepares the mailbox with the Buffer write command value, the buffer address and the buffer size. The host then forces the device in Applet mode using a TCL_Go command. The host polls the end of payload programming by trying to read the state value in the mailbox. The device will answer to the host as soon as it returns to SAM-BA monitor mode. In case of USB connection, when the host polls while the device is in Applet mode, the device NACK IN packets sent by the host. Applet execution has to be short enough in order to prevent from connection timeout error. In case of long programming or erasing operation, from time to time, the device shall leave Applet mode to return to SAM-BA monitor mode to be able to achieve the current pending host TCL_ReadInt command within the timeout threshold.

Figure 3-2. Applet Workflow

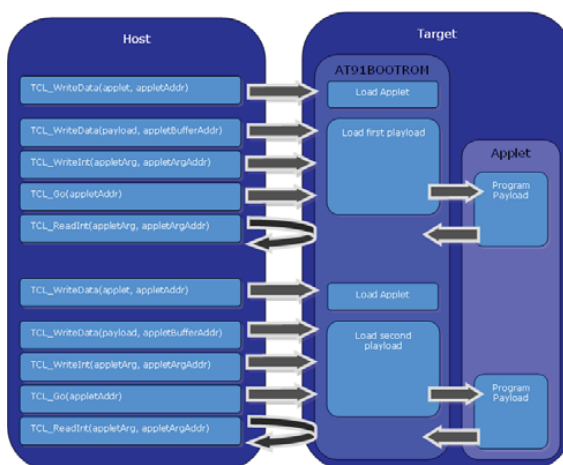
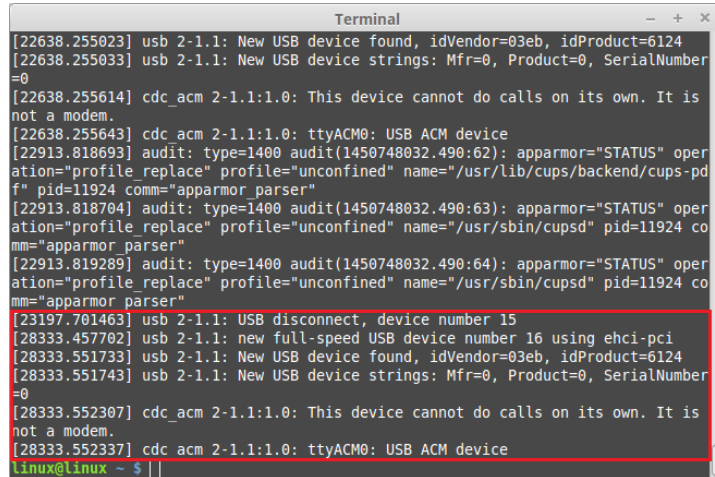


Figure 3-4. Device Enumeration



```
[22638.255023] usb 2-1.1: New USB device found, idVendor=03eb, idProduct=6124
[22638.255033] usb 2-1.1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[22638.255614] cdc_acm 2-1.1:1.0: This device cannot do calls on its own. It is not a modem.
[22638.255643] cdc_acm 2-1.1:1.0: ttyACM0: USB ACM device
[22913.818693] audit: type=1400 audit(1450748032.490:62): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/lib/cups/backend/cups-pdf" pid=11924 comm="apparmor_parser"
[22913.818704] audit: type=1400 audit(1450748032.490:63): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/sbin/cupsd" pid=11924 comm="apparmor_parser"
[22913.819289] audit: type=1400 audit(1450748032.490:64): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/sbin/cupsd" pid=11924 comm="apparmor_parser"
[23197.701463] usb 2-1.1: USB disconnect, device number 15
[28333.457702] usb 2-1.1: new full-speed USB device number 16 using ehci-pci
[28333.551733] usb 2-1.1: New USB device found, idVendor=03eb, idProduct=6124
[28333.551743] usb 2-1.1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[28333.552307] cdc_acm 2-1.1:1.0: This device cannot do calls on its own. It is not a modem.
[28333.552337] cdc_acm 2-1.1:1.0: ttyACM0: USB ACM device
linux@linux ~ $
```

As SAM-BA uses scan function in `sam-ba.dll` to determine all devices connected to the PC, it is recommended to connect the target to the PC before launching SAM-BA. When SAM-BA starts, a pop-up window appears, which allows users to choose the connection and board.

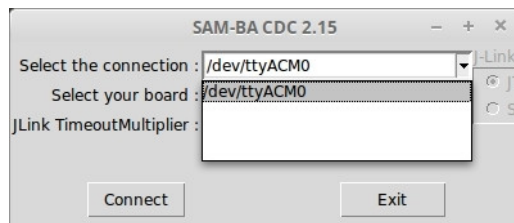
To connect to SAM-BA GUI,

1. Launch SAM-BA with the superuser privileges.

```
sudo sam-ba
```

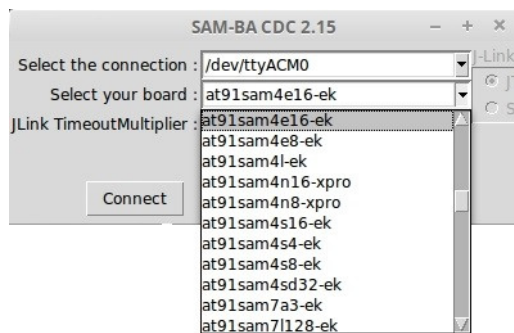
2. Select connection in **Select the connection** list.

Figure 3-5. Selecting Connection



3. Select the board from the board list and press **Connect**. SAM-BA GUI window should pop-up.

Figure 3-6. Selecting Device



3.3. Understanding SAM-BA GUI

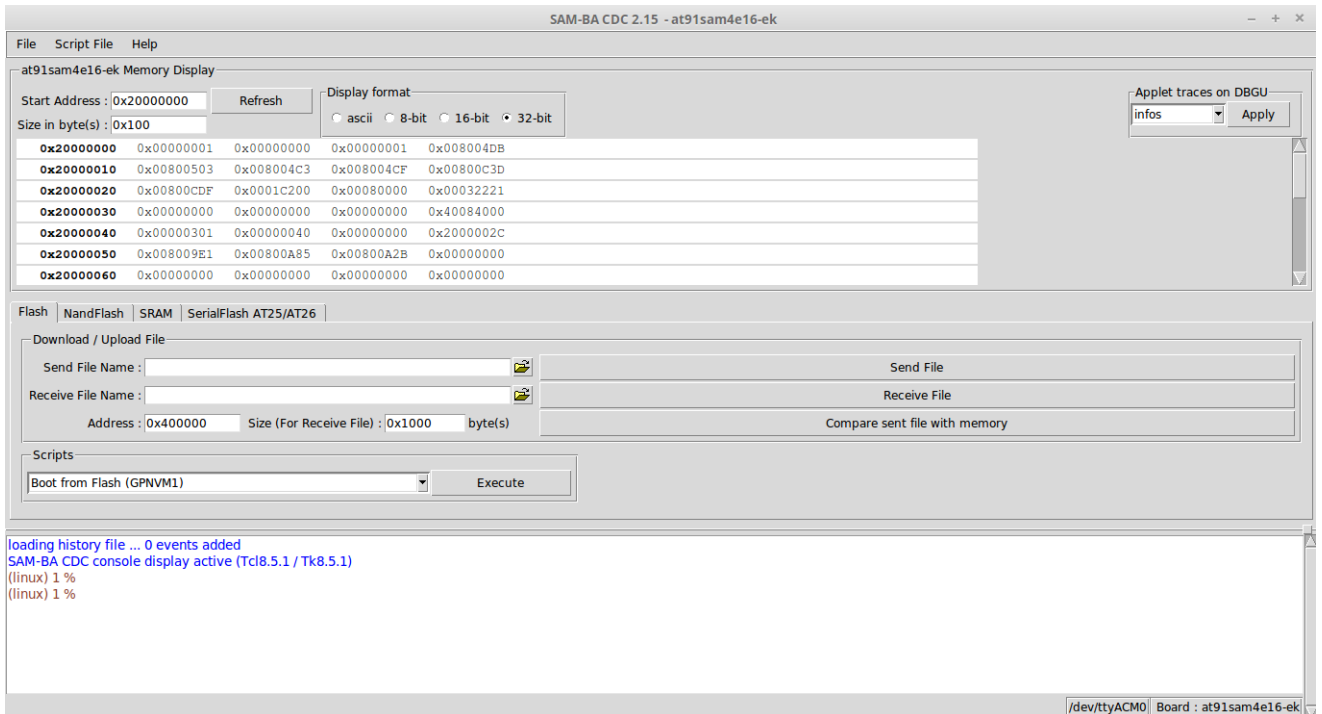
When SAM-BA is launched, after selection of the board and the communication link, the main window appears.

It contains three different areas:

1. Memory Display area
2. Memory Download area
3. TCL Shell area

The Memory Display area and the Memory Download area are used to simplify the memory access.

Figure 3-7. SAM-BA Main Window



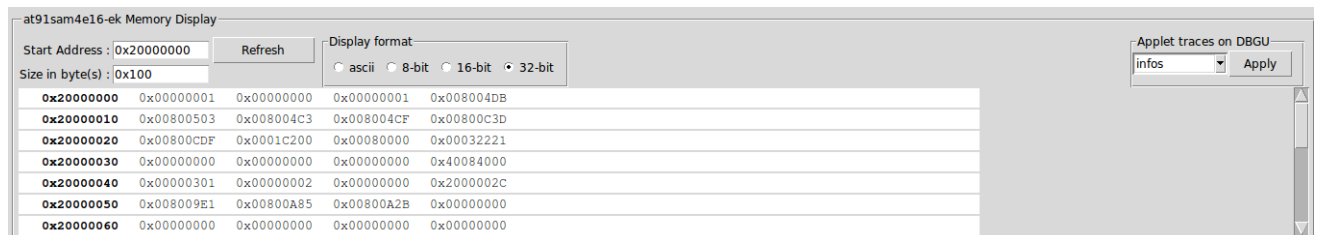
The TCL shell area is a standard TCL shell. All the text typed in the shell is interpreted by a TCL interpreter. This area provides access to TCL commands. Type “puts Welcome” and the result is “Welcome”. Type “expr 3 + 7” and the result is “10”.

3.3.1. Memory Display Area

In this area, user can display a part of the microcontroller memory content. Three different display formats can be used: 32-bit word, 16-bit half-word, or 8-bit byte, with a maximum display of 1024-byte long memory area. Values can also be edited by double-clicking on them.

Note: Only valid memory areas or system/user peripheral areas are displayed. If a forbidden address is supplied or if a memory overrun occurs, an error message is written in the TCL Shell area .

Figure 3-8. Memory Display Area



3.3.1.1. Read Memory

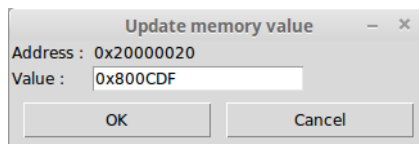
1. Enter the address of the area that the user want to read in the starting address field.
2. Enter the size of the area to display.

3. Choose display format: 32-bit word, 16-bit half-word or 8-bit byte. This automatically refreshes the contents of the memory.
4. Press the **Refresh** button.

3.3.1.2. Edit Memory Content

A few of the memories and/or embedded peripherals can be edited.

Figure 3-9. Edit Memory Content

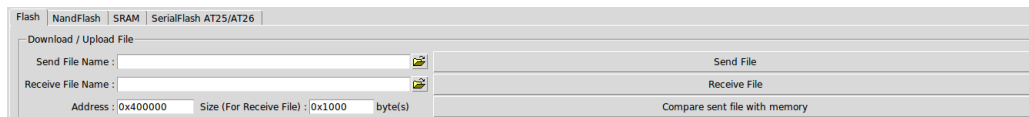


1. Double-click on the value that users want to update in an editable pop-up window.
Note: Only some memories can be updated in this method such as static RAM or SDRAM (if previously initialized). If the users try to write into the other memory types, command is ignored.
2. Press **OK** to update the value in the Memory display area. The corresponding TCL command is displayed in the TCL Shell area.
Note: Only the lowest bits of the value are considered if the format of the value entered is higher than the display format.

3.3.2. Memory Download Area

The Memory download area provides a simple way to upload and download data. For each memory, files can be sent and received and the target's memory content can be compared with a file on the computer. This area also provides access to some specific scripts for the different on-board memories such as SRAM, SDRAM, Flash, DataFlash, and NandFlash.

Figure 3-10. Memory Download Area



3.3.2.1. Flashing Firmware

Select the specific memory by clicking on the corresponding tab.

1. Specify the file name and location in the Send File name field or open the file browser by clicking on the **Open Folder** button and select it. If the users enter a wrong file name or format, an error message will be displayed in the TCL Shell.
2. Enter the destination address in the selected memory where the file must be written. If users enter a forbidden address, or if the file overruns the memory size, an error message will be displayed in the TCL Shell.
Note: A forbidden address corresponds to an address outside the selected memory range address. Send the file using the **Send File** button. Ensure that the memory is initialized correctly before sending any data.

3.3.2.2. Reading From Memory

Select the memory by clicking on the corresponding tab.

1. Enter the file name location in the **Receive File name** field or open the file browser by clicking on the **Open Folder** button and select it. If the user enters a wrong file name, an error message is displayed in the TCL Shell.
2. Enter the address of the first data to read in the **Address** field.
3. Enter the data size to read in the **Size** field.

4. If the users enter a forbidden address or if their file size overruns the memory size, an error message will be displayed in the TCL Shell.

Note: A forbidden address corresponds to an address outside the selected range of memory addresses. Get data using **Receive File** button. Ensure that your memory is initialized correctly before receiving any data.

3.3.2.3. Compare Memory with a File

Usually, this feature allows to verify if a file is stored correctly into the memory. But users can compare any files with the content stored in the memory. The comparison is based on the size of the selected file.

1. Select the specific memory by clicking the corresponding tab.
2. Enter the file name location in the **Send File name** field or open the file browser by clicking on the **Open Folder** button and select it. If the users enter a wrong file name, an error message will be displayed in the TCL Shell.
3. Enter the address of the first data to compare with the selected file in the **Address** field. If the users enter a forbidden address, or if the file size overruns the memory size, an error message will be displayed in the TCL Shell.

Note: A forbidden address corresponds to an address outside the selected memory range address. Compare the selected file with the memory content using the **Compare sent file with memory** button. If the file size matches the file size of the memory content, a message box is displayed. Ensure that the memory is correctly initialized before comparing any data.

3.3.3. Script File Functionality

SAM-BA allows users to create, edit and execute script files. A script file configures the device easily or automatically runs significant scripts. The **Script File** menu supplies commands to start and stop recording, to execute, reset, edit and save the recording file. The name of the generated file is `historyCommand.tcl`. The command sequence in `historyCommand.tcl` can be used to create a custom script.

3.3.3.1. Start/Stop/Reset Recording

In the **Script File** menu, select **Start Recording** to begin the record. Now, all the commands to be executed in different blocks of the software are recorded in a specific file called `historyCommand.tcl`. This file is located in the `usr` directory.

Note: This file can only be written through the Start/Stop/Reset Recording commands. If the recording file is not reset, the new recorded commands will be added at the end of the `historyCommand.tcl` file. When users want to stop recording, select **Stop Recording** in the menu. If they want to erase the contents of `historyCommand.tcl`, select **Reset Recording**.

3.3.3.2. Edit the Script File

The users have the functionality to edit `historyCommand.tcl`. Use the **Edit Script File** command in the **Script File** menu. A new window appears in which users can edit and save the content. The users can save their modified script in another file through the **Save file** button. Thus several configuration scripts for specific use are available.

3.3.3.3. Execute the Script File

SAM-BA can execute the TCL script files directly from a shell without using the GUI. There are two ways to executing a script file.

1. Run SAM-BA for more information on how to execute TCL script files from a shell.
2. Use the **Execute Script File** command in the GUI **Script File** menu and enter the TCL file to execute. Messages that notice the correct execution of the script are displayed in the TCL Shell and/or through message boxes.

Note: All TCL commands can be executed using script files.

3.4. SAM-BA in command line

SAM-BA can operate in a graphical mode or it can be launched in command line mode with a TCL script in parameter. Both modes can be easily combined together to easily obtain a powerful loading solution on SAM devices customized for the current project. To use the command line, type the following in a shell:

```
sam-ba [Communication Interface] [Board] [Script_File][Script_File Args][> log file ]
```

where:

- [Communication Interface]: **\dev\ttyACM0**
- [Board]: Name of the board accessible through the SAM-BA connection window
- [Script_File] (Optional): Path to the TCL Script File to execute
- [Script_File Args] (Optional): TCL Script File Arguments
- [>logfile [2>&1]] (Optional): redirect STDOUT and/or STDERR to a file

For example:

```
sam-ba \usb\ttyACM0 at91sam4e16-ek myCommand.tcl > logfile.txt 2>&1
```

Note:

1. If the users enter a bad argument in the command line or if there are communication problems, SAM-BA will not be able to start.
2. On a 64-bit machine sam-ba_64 binary should be used in place of sam-ba.

4. Tested Devices

Table 4-1. Tested Devices

S.No.	Tested Kit	Device
1	SAM V71 Xplained	ATSAMV71Q21
2	SAM 4E Xplained Pro	ATSAM4E16E
3	SAM4S-EK2	ATSAM4SD32C
4	SAM4E-EK	ATSAM4E16E
5	SAM3S-EK	ATSAM3S4C
6	SAM3X-EK	ATSAM3X8H
7	SAM4S XPLAINED	ATSAM4S16C
8	SAM3U-EK	ATSAM3U4E
9	SAM D21 Xplained Pro	ATSAMD21J18A

5. FAQ

1. My device selection for SAM-BA applet is greyed out, what should I do?

If the device selection scroll of SAM-BA is greyed out, this means that SAM-BA was not able to detect any USB connection

Ensure that:

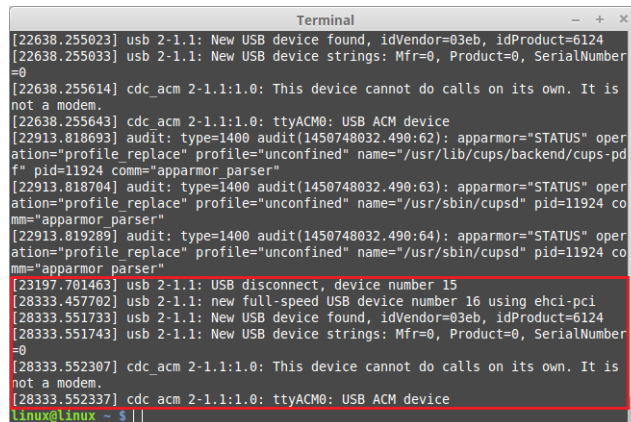
- SAM-BA monitor is running over the device.
- Kernel libraries are updated.
- SAM-BA is running under super user privileges.

2. How to ensure Kernel is up-to-date to support ACM drivers?

To ensure that kernel libraries are up-to-date connect the device running SAM-BA monitor, open the terminal and type:

```
dmesg
```

Figure 5-1. Kernel Log



```
Terminal
[22638.255023] usb 2-1.1: New USB device found, idVendor=03eb, idProduct=6124
[22638.255033] usb 2-1.1: New USB device strings: Mfr=0, Product=0, SerialNumber
=0
[22638.255614] cdc_acm 2-1.1:1.0: This device cannot do calls on its own. It is
not a modem.
[22638.255643] cdc_acm 2-1.1:1.0: ttyACM0: USB ACM device
[22913.818693] audit: type=1400 audit(1450748032.490:62): apparmor="STATUS" oper
ation="profile_replace" profile="unconfined" name="/usr/lib/cups/backend/cups-pd
f" pid=11924 comm="apparmor_parser"
[22913.818704] audit: type=1400 audit(1450748032.490:63): apparmor="STATUS" oper
ation="profile_replace" profile="unconfined" name="/usr/sbin/cupsd" pid=11924 co
mm="apparmor_parser"
[22913.819289] audit: type=1400 audit(1450748032.490:64): apparmor="STATUS" oper
ation="profile_replace" profile="unconfined" name="/usr/sbin/cupsd" pid=11924 co
mm="apparmor_parser"
[23197.701463] usb 2-1.1: USB disconnect, device number 15
[28333.457702] usb 2-1.1: new full-speed USB device number 16 using ehci-pci
[28333.551733] usb 2-1.1: New USB device found, idVendor=03eb, idProduct=6124
[28333.551743] usb 2-1.1: New USB device strings: Mfr=0, Product=0, SerialNumber
=0
[28333.552307] cdc_acm 2-1.1:1.0: This device cannot do calls on its own. It is
not a modem.
[28333.552337] cdc_acm 2-1.1:1.0: ttyACM0: USB ACM device
linux@linux ~ $
```

If the line "This device cannot do calls on its own. It is not a modem." is present that indicates that the Kernel is up-to-date.

3. How can I update Kernel libraries to support SAM-BA?

To update the kernel, open the terminal and type the following code:

```
sudo apt-get install linux-image-generic linux-headers-generic
```

This will update the kernel image.

6. References

1. SAM-BA download page: <http://www.atmel.com/tools/atmelsam-bain-systemprogrammer.aspx>
2. Linux for SAM website: <http://www.at91.com/linux4sam/bin/view/Linux4SAM/>
3. SAM-BA for SAM4L: http://www.atmel.com/images/atmel-42051-sam-ba-for-sam4l_application-note_at03454.pdf
4. SAM-BA Bootloader for SAM D21: http://www.atmel.com/Images/Atmel-42366-SAM-BA-Bootloader-for-SAM-D21_ApplicationNote_AT07175.pdf
5. SAM-BA Bootloader for SAMD20: http://www.atmel.com/images/atmel-42238-uart-based-sam-ba-bootloader-for-sam-d20_ap-note_at04189.pdf

7. Revision history

Doc. Rev	Date	Comments
42728A	08/2016	Initial document release



Atmel® | Enabling Unlimited Possibilities®



Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA T: (+1)(408) 441.0311 F: (+1)(408) 436.4200 | www.atmel.com

© 2016 Atmel Corporation. / Rev.: Atmel-42728A-Using-SAM-BA-for-Linux-on-SMART-ARM-based-Microcontrollers_AT15004_Application Note-08/2016

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, SAM-BA® and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected®, and others are registered trademarks of ARM Ltd. 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.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.