Application Report
SPRACC3–January 2018

Flashing Utility - mflash

Prawal Gangwar *ADAS Software*

ABSTRACT

This application report focuses mainly on the procedure to flash the Secondary BootLoader and AppImage into TDA3xx Systems. sbl_mflash algorithms are not in the scope of this document. But the procedure to configure and build the executable is defined here.

Contents

1 Introduction 1
2 Need of Multicore Flashing Utility 1
3 Brief Overview of its Working 2
4 System/Software Setup 2
5 Flashing TDA3xx via UART Interface 6
6 mflash Algorithm 8

List of Figures

1 Tera Term Setup Window 2
2 Serial Output When Booting in UART Boot Mode 3
3 minicom Serial Communication Configuration 4
4 Serial Output in Minicom on Linux 5
5 Serial Output on Minicom (Zoomed) 5

List of Tables

1 Serial Port Configuration 4

Trademarks

1 Introduction

This application report provides detailed procedure for flashing the binary images to QSPI Flash memory using the Universal Asynchronous Receiver/Transmitter (UART) interface for the TDA2xx and TDA3xx Boards. Generally, the MMC/SD boot mode can be used to boot the fresh production board/EVM. In case there is not an external MMC/SD card available as part of production EVM or final product, this application report will be useful to flash the images to the factory boards using the UART boot mode of TDA3xx, respectively.

2 Need of Multicore Flashing Utility

Typically, there are multiple interfaces to boot from when it is about Evaluation Module (EVMs) but for production system/target boards at customer places, interfaces are very limited. Thus, in order to make it easy and fast for customer to flash binaries to production device a Windows/Linux based utility is needed. mflash currently supports flashing on TDA3x systems via UART but could be extended similarly for other platforms, too.

3 Brief Overview of its Working

- ROM bootloader runs and checks the sysboot switch settings to find peripheral boot mode
- Receives the peripheral boot mode request instruction via UART3
- Receives the sbl_mflash via UART3 and puts it into the on-chip memory
- sbl_mflash starts to execute and interacts with the PC to download the sbl and AppImage into the Flash memory.

4 System/Software Setup

4.1 System Requirements

The executable was created and tested on:

- Windows 10 based 64 bit PC
- Ubuntu 16.04.2 LTS 64 bit PC

4.2 Windows

4.2.1 Finding the Correct COM Port Using Tera-Term

Tera term is one of the serial port communication application for windows which can be downloaded and installed. Connect a serial cable to the UART port of the EVM and the other end to the serial port of the PC. It will detect four UART ports; you need to select the third Port of EVM with the following settings that can be setup from Setup → Serial Port to the following:

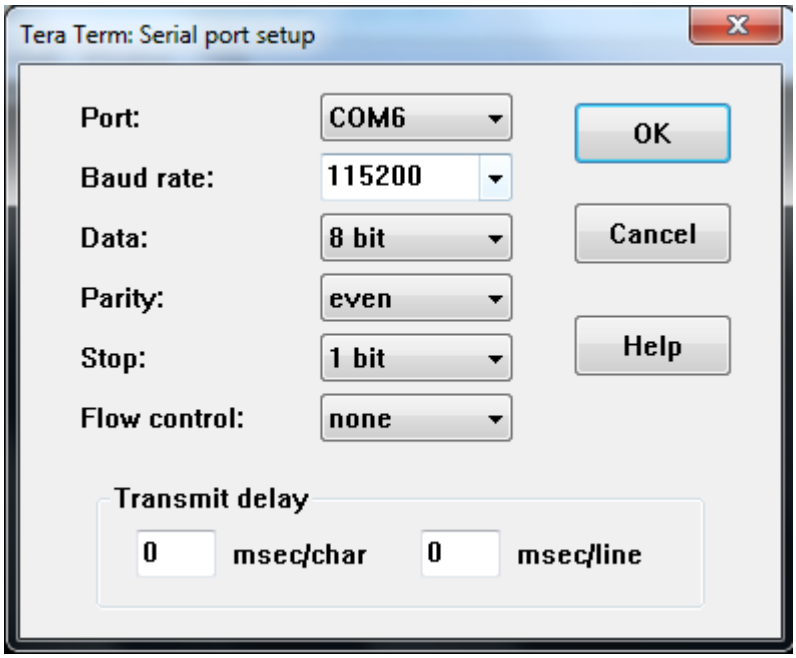

Figure 1. Tera Term Setup Window

To find the correct COM Port number for the peripheral boot put the board in UART boot mode by changing the SYSBOOT switch SW2 to [00010000][10000001] for TDA3xx board.

With correct settings, it should continuously display AL! on the TeraTerm.

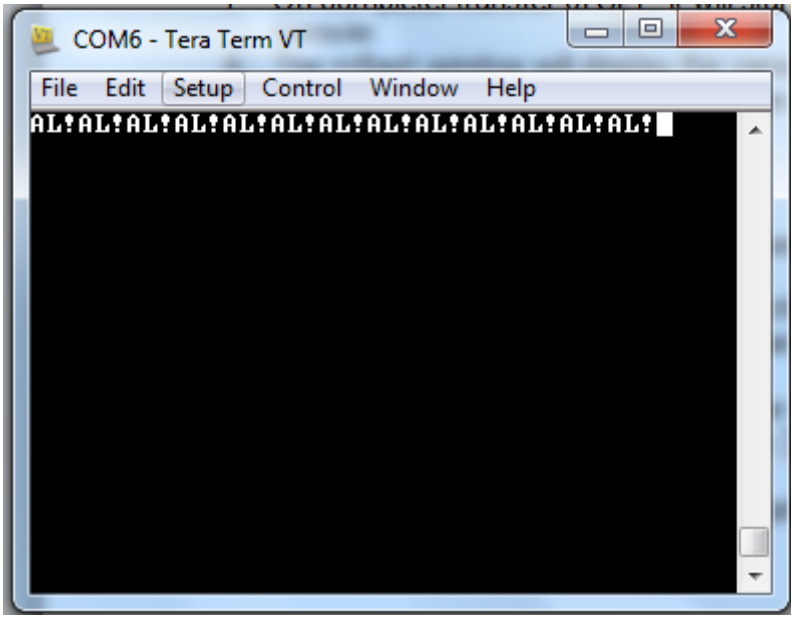

Figure 2. Serial Output When Booting in UART Boot Mode

Note the COM Port number. For example if COM6 displays AL! then the COM Port number is 6. This is also an assurance that the switch settings on the EVM are correct. Disconnect and close TeraTerm.

4.2.2 Configuring the Flashing Script

Since the flashing tool takes all the parameters via command line, there is a configuration script in the same folder mflash_run_config.bat that can be modified to flash every time. The first 6 lines can be changed according to the usage.

Its structure is as follows:

```
SET sbl_mflash=<path_to_sbl_mflash>
SET appimage_location=<path_to_AppImage>
SET appimage_offset=<AppImage offset>
SET sbl_location=<path_to_sbl>
SET sbl_offset=<sbl offset>
SET uart_port_number=<port_number>
```

A typical example where all the files are in the same folder as the mflash executable would be:

```
SET sbl_mflash="sbl_mflash"
SET appimage_location="AppImage_BE"
SET appimage_offset="0x80000"
SET sbl_location="sbl"
SET sbl_offset="0x00"
SET uart_port_number="5"
```

The last line executes the mflash executable with the above given parameters:

```
mflash -M %sbl_mflash% -P %uart_port_number% -F %appimage_location% %appimage_offset% -
F %sbl_location% %sbl_offset% -C
```

4.3 Linux

4.3.1 Finding the Correct COM Port Using Minicom

Use minicom on Linux to find the correct COM port. Ports are usually named as /dev/ttyUSBx where x is the port number required. Usually it is the third port on the list. One way to look at all the serial communication devices connected to the system is to look into the /dev/ directory and the ports would appear as files named as /dev/ttyUSBx where x is the required port number.

```
$ cd /dev/
$ ls
```

In the list of connected serial ports, the third port is required port ID. It might be different on different systems, so check it once to be sure.

Configure the Minicom Serial Port with the following setting:

Table 1. Serial Port Configuration

Parameter	Value
Baud	115200
Data	8 bit
Parity	Even
Stop Bit	1 bit
Flow Control	None

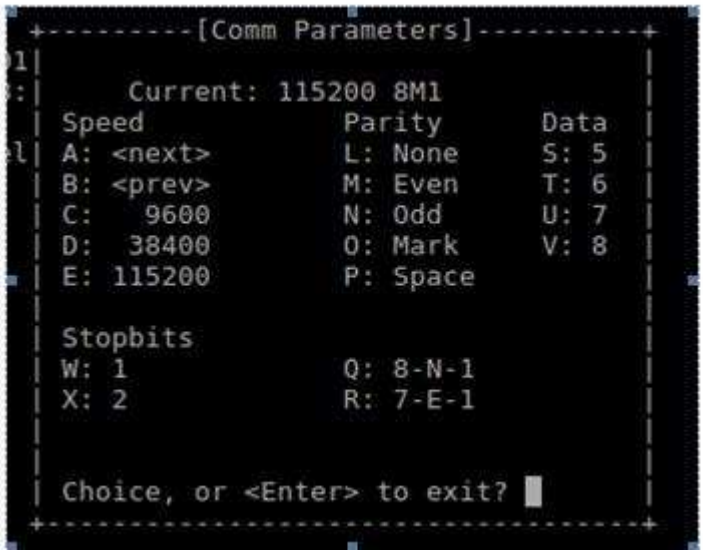

Figure 3. minicom Serial Communication Configuration

When connected to the correct COM Port with the proper switch settings, the terminal should continuously display the printable part of the ASIC id which should be "AL!". Also minicom also prints the non-printable characters so AL! is visible but along with other characters.

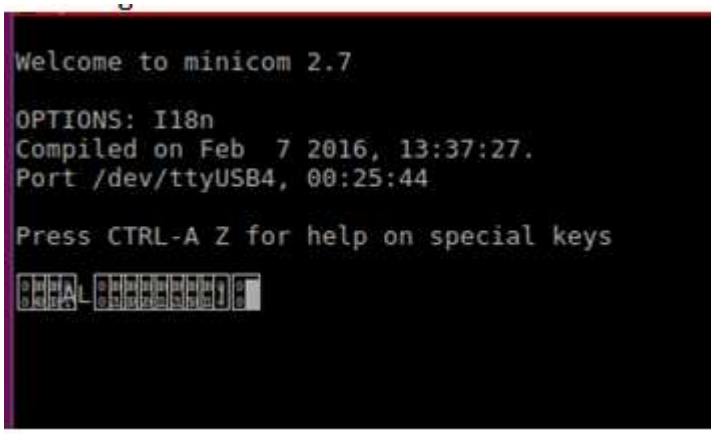

Figure 4. Serial Output in Minicom on Linux

If zoomed enough, it looks correct:

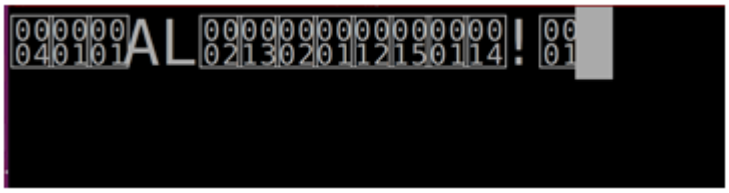

Figure 5. Serial Output on Minicom (Zoomed)

Note the Port number that displays the correct id. This is also an assurance that the switch settings on the EVM are correct.

4.3.2 Configuring the Flashing Script

Since the flashing tool takes all the parameters via command line, there is a configuration script that can be modified to flash every time. The first six lines can be changed according to the usage.

Its structure is as follows:

```
sbl_mflash=<path_to_sbl_mflash>
appimage_location=<path_to_AppImage>
appimage_offset=<AppImage offset>
sbl_location=<path_to_sbl>
sbl_offset=<sbl offset>
uart_port_number=<port_number>
```

A typical example where all the files are in the same folder as the mflash executable would be:

```
sbl_mflash="sbl_mflash"
appimage_location="AppImage_BE"
appimage_offset="0x80000"
sbl_location="sbl"
sbl_offset="0x00"
uart_port_number="4"
```

The last line executes the mflash executable with the above given parameters:

```
sudo ./mflash -M $sbl_mflash -P $uart_port_number -F $appimage_location $appimage_offset -
F $sbl_location $sbl_offset -C
```

4.4 Building sbl_mflash

To build sbl_mflash (sbl that is sent by PC side mflash utility to ROM code over uart):

1. Go to pdk/packages/ti/build
2. Use following build command

```
$ make -s -j sbl BOARD=tda3xx-evm BOOTMODE=uart SBL_TYPE=mflash
```

3. Go to pdk/packages/ti/boot/sbl_auto/tools/mflash
   a. Ensure paths and profile mentioned in the script is right
   b. Execute sbl_mflash_create_tda3xx.sh / .bat
4. On success pdk/packages/ti/boot/sbl_auto/tools/mflash/mflash_tda3xx directory will be created.
5. sbl_mflash is created in mflash_tda3xx.

4.5 Building AppImage for TDA3xx and TDA2xx

The procedure to build AppImage and sbl_qspi can be done as described in VisionSDK_UserGuide_TDAxxx.pdf.

4.6 Building mflash for TDA3xx

To build mflash in any environment a gcc compiler is required. Go to the directory pdk/packages/ti/boot/sbl_auto/tools/mflash and run the following command to build the executable named “mflash”.

```
$ gcc -o mflash mflash_uart.c
```

The executable mflash will be created in the same directory.

5 Flashing TDA3xx via UART Interface

To Flash sbl and AppImage in TDA3xx in Peripheral Boot Mode:

1. Build sbl_mflash and mflash for TDA3xx if not already built as described above.
2. Make the following connections:
   a. Connect a serial cable to the UART port of the EVM and the other end to the serial port of the PC
   b. Change the sysboot switches SW2[0:7] and SW3[8:15] to [00010000][100000001] and SW8 to [01000001]
   c. Connect and Power Reset the board SW4
3. Note the appropriate COM Port as described previously.
4. Reset the board.
5. 5. Run the mflash_run_config.sh(.bat) file after providing appropriate parameters, or provide your own command in the following syntax:

```
$ ./mflash -M <sbl_mflash_address>                [Required]   {to give sbl_mflash path}
           -P <port_number>                       [Required]   {UART3 port_number}
           -F <file_location> <file_offset>       [Required]   {file location and offset}
           -F <file_location> <file_offset>       [Optional]
           -C                                     [Optional]   {to clear the flash memory}
```

   For example, mflash -M “sbl_mflash” -P “5” -F “AppImage_BE” “0x80000” -F “sbl” “0x00” -C.
6. Put the board in qspi boot mode (change the switch settings) and restart it to boot the sbl and AppImage from flash.

NOTE:

- Note that AppImage and sbl here should be in Big Endian (BE).
- Root permission/admin rights might be required in several cases.

5.1 Sample Logs

```
[PC] File          0       AppImage_BE
[PC] Offset        0       0x80000
[PC] File          1       sbl
[PC] Offset        1       0x00
[PC] com \\.\COM5
[PC] ##########Starting USB/UART Flasing Utility##############
[PC] Put UART Boot Mode, make fresh UART connection & restart
[PC] Press Enter when done...

     Baud     = 115200
     Parity   = 2
     StopBits = 0
     ByteSize = 8
[PC] Opening serial port successful
[RBL]4  [RBL]1  [RBL]5  [RBL]1  [RBL]41 [RBL]4c [RBL]7  [RBL]2  [RBL]13 [RBL]2  [RBL]1  [RBL]0
[RBL]12 [RBL]15 [RBL]1  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[RBL]0  [RBL]0  [RBL]0    [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]14
[RBL]21 [RBL]1  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0    [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[PC] Requesting the ASIC ID
[RBL]4  [RBL]1  [RBL]5  [RBL]1  [RBL]41 [RBL]4c [RBL]7  [RBL]2  [RBL]13 [RBL]2  [RBL]1  [RBL]0
[RBL]12 [RBL]15 [RBL]1  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[RBL]0  [RBL]0  [RBL]0    [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]14
[RBL]21 [RBL]1  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0    [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0  [RBL]0
[PC] Requesting PERI_REQ mode
[PC] Sending SBL_MFLASH filesize.
[PC] Size of sbl_mflash = 105866
[PC] Sending SBL_MFLASH... Please wait
[PC] File Size  = 105866
[PC] ##
[PC] Transfer Complete. Time = 10.000
[PC] Opening port for sbl_mflash.
     Baud     = 12000000
     Parity   = 0
     StopBits = 0
     ByteSize = 8
[PC] Opening serial port successful.
[PC] sbl_mflash switch On Request Sent.
[TDAxx] Utility mflash will Execute now.
[TDAxx] Setting up QSPI
[TDAxx] QSPI Spansion 4 bit Device type
[TDAxx] MID - 0x1
[TDAxx] DID - 0x18
[TDAxx] !!______TDAxx flashing utility______!!1
[TDAxx] Erasing entire QSPI Flash..This takes 50-60 seconds.
0x008DAxx] Erase Completed!!!2
[PC] Download started[PC] File Size  = 52368
[PC] #
[PC] Write File Completed.

0x80000me taken to download file  = 0.002
[PC] Download started[PC] File Size  = 6160892
[PC] ##################################################################################################
[PC] Write File Completed.

[PC] Time taken to download file  = 17.003
[TDAxx] Exiting.
[PC] ###############!!!!mflash shutting down!!!!##############
```

6 mflash Algorithm

1. Read the ASIC id from the ROM Bootloader (RBL)
2. Request the Peripherial Boot Mode
3. Send the sbl_mflash filesize
4. Send the sbl_mflash
5. Close the port and reopen it with UART settings of the sbl_mflash
6. Perform the two-way handshake
   a. sbl_mflash as soon as it boots up will start sending character ‘a’ continuously
   b. mflash.exe will wait for the character ‘a’ and upon receiving will send character ‘1’
   c. sbl_mflash upon receiving ‘1’ will send out character ‘b’ and get ready to proceed
   d. mflash.exe upon receiving ‘b’ will get ready to transfer the files
7. sbl_mflash sends a request for the command.
8. mflash reads the request and sends the required actions like sending the filesize, file, offset, clear the flash, and so forth.
9. Upon completing the previous request, sbl_mflash waits for another request

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265