Flashing Utility - mflash

Prawal Gangwar *ADAS software*

Abstract

This application report provides detailed procedure for flashing the binary images to QSPI Flash memory using the 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 TDAxx based devices.

Need of Multicore Flashing Utility

Typically, we have multiple interfaces to boot from when it is about EVMs (Evaluation Module) 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, TDA2x systems via UART but could be extended similarly for other platforms too.

Scope

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

Brief overview of its working

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

System/Software Setup

System Requirements:

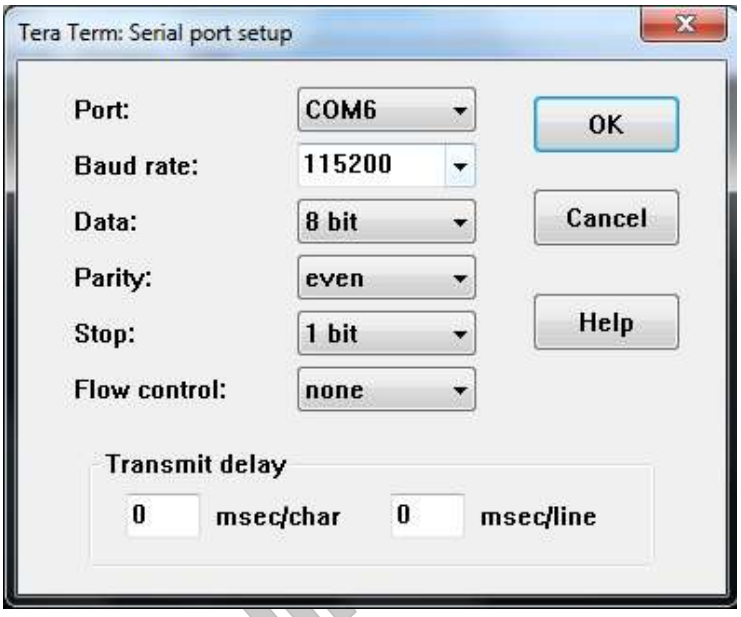
The executable was created and tested on

- Windows 10 based 64 bit PC
- Ubuntu 18.04 LTS 64 bit PC

Windows

Finding the correct COM Port using Tera-term

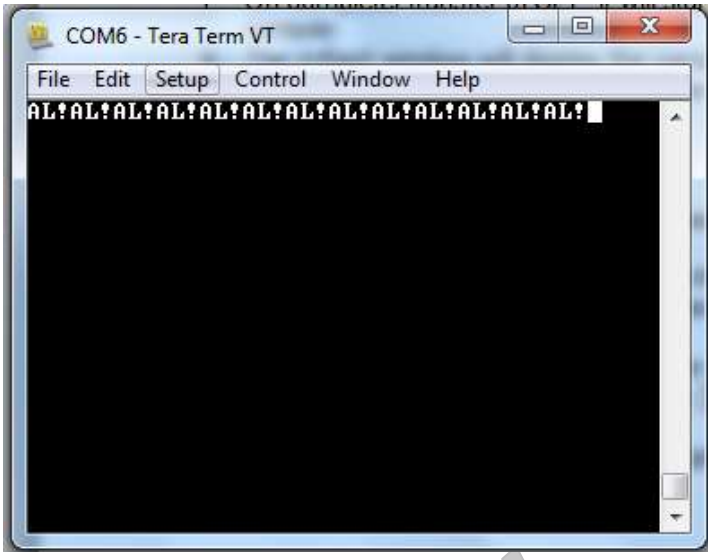
Tera term is one of the serial port communication applications for windows which can be downloaded and installed. Connect a serial cable to the UART port of the EVM and the other end to the serial port of the PC. It will detect multiple UART ports; you need to select the UART3 Port of EVM with the following settings which can be setup from Setup -> Serial Port to the following:



To find the correct COM Port number for the peripheral boot put the board in UART boot mode by changing the SYSBOOT switch

- For TDA3xx-emv, change the switch SW2[0:7], SW3[8:15] to [00010000][100000001] and SW8 to [01000001]
- For TDA2px-evm, change the switch SW3[0:7], SW4[8:15] to [11001000][100000001] and SW6 to [00000000]
- For TDA2xx-evm, change the switch SW3[0:7], SW4[8:15] to [11001000][100000001] and SW5 to [00010000]
- For TDA2ex-evm, change the switch SW3[0:7] and SW4[8:15] to [11001000][100000001]
- For TDA3xx-rvp, no sys-boot settings are required

With correct settings, it should continuously display AL! on the TeraTerm for TDA3xx. The printed characters may vary for different platforms. UART3 is not enabled on many boards. This can be enabled by a simple register modification on the evm. By default UART3 is only populated on TDA2xx-evm and TDA3xx-evm



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.

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* which can be modified to flash every time. First 6 lines can be changed according to the usage.

Its structure is as follows:

```
SET sbl_mflash=<path_to_sbl_mflash>
SET board=<board>
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_tda2ex-evm"
SET board="tda2exevm"
SET appimage_location="AppImage_BE"
SET appimage_offset="0x80000"
SET sbl_location="sbl"
SET sbl_offset="0x00"
SET uart_port_number="34"
```

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

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

Linux

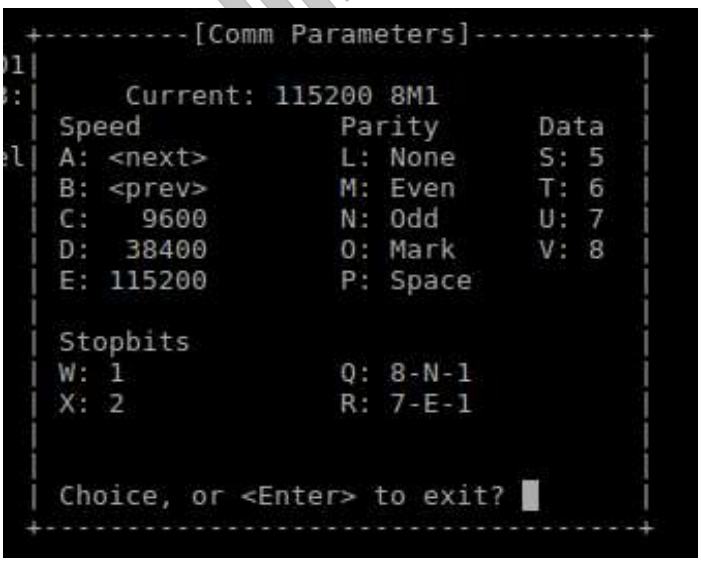
Finding the correct COM Port using minicom

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.

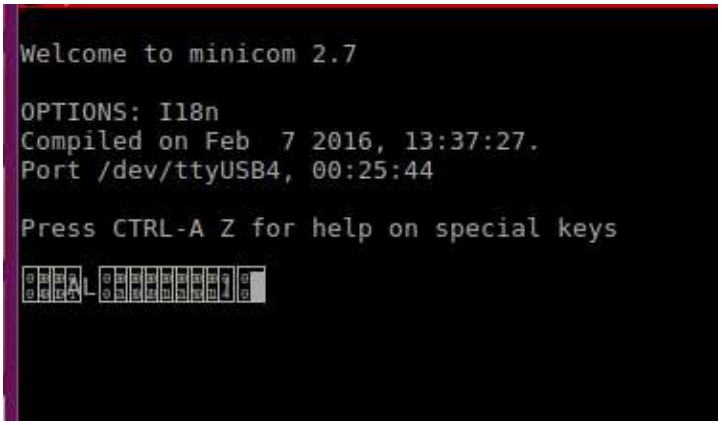
$ cd /dev/
$ ls

Configure the Minicom Serial Port with the following setting:

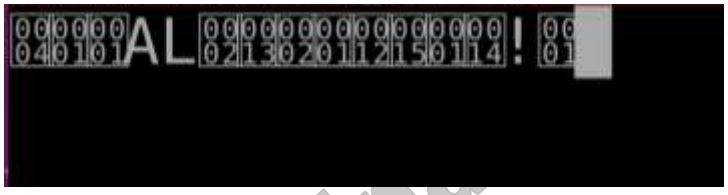
Parameter	Value
Baud	115200
Data	8 bit
Parity	Even
Stop Bit	1 bit
Flow Control	None



When connected to the correct COM Port with the proper switch settings the terminal should continuosly 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. The printed characters may vary for different platforms.



If zoomed enough, it looks correct:



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

Configuring the flashing script

Since the flashing tool takes all the parameters via command line, there is a configuration script which can be modified to flash every time. First 6 lines can be changed according to the usage.

Its structure is as follows:

```
sbl_mflash=<path_to_sbl_mflash>
board=<board>
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_tda2ex-evm"
board="tda2exevm"
appimage_location="AppImage_BE"
appimage_offset="0x80000"
sbl_location="sbl"
sbl_offset="0x00"
uart_port_number="14"
```

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

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

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 for tda3xx-evm
   ```
   $ make -s -j sbl BOARD=tda3xx-evm BOOTMODE=uart SBL_TYPE=mflash
   ```
   Use following build command for tda2px-evm
   ```
   $ make -s -j sbl BOARD=tda2px-evm CORE=a15_0 BOOTMODE=uart SBL_TYPE=mflash
   ```
   Use following build command for tda2ex-evm
   ```
   $ make -s -j sbl BOARD=tda2ex-evm CORE=a15_0 BOOTMODE=uart SBL_TYPE=mflash
   ```
   Use following build command for tda2xx-evm
   ```
   $ make -s -j sbl BOARD=tda2xx-evm CORE=a15_0 BOOTMODE=uart SBL_TYPE=mflash
   ```
   Use following build command for tda3xx-rvp
   ```
   $ make -s -j sbl BOARD=tda3xx-rvp 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.sh / .bat <soc_name> <profile> <boardType>*
   e.g. `sbl_mflash_create.sh tda3xx debug evm`
4. On success *pdk/packages/ti/boot/sbl_auto/tools/mflash/sbl_mflash_tda3xx_evm* will be created.

Building AppImage for TDA3xx and TDA2xx

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

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

Flashing TDAxx via UART interface

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

1. Build sbl_mflash and mflash for TDAxx if not already built as described above.
2. Make the following connections:

- Connect a serial cable to the UART port of the EVM and the other end to the serial port of the PC
- Change the sysboot switches as defined before
- Connect and Power Reset the board

3. Note the appropriate COM Port as described previously.
4. Reset the board
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}
-S <board>	[Required]	
-F <file_location> <file_offset>	[Required]	{file location and offset}
-F <file_location> <file_offset>	[Optional]	
-C	[Optional]	{to clear the flash memory}

e.g. 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:

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

APPENDIX

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 [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 [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!!!!###############

mflash Algorithm

1. Read the ASIC id from the RBL(ROM Bootloader)
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 handshake sequence continuously
    b. mflash.exe will wait for the sequence and upon receiving will send a response
    c. sbl_mflash upon receiving the response will send out the second response and get ready to proceed
    d. mflash.exe upon receiving the second response will get ready to transfer the files

7. sbl_mflash will request for files/data by sending relevant requests.
8. mflash will read the request and send the perform the required actions like sending the filesize, file, offset, clear the flash, etc.
9. sbl_mflash upon completing the previous request will wait for another request.