Aerospace & Defence | Lighting | Power Management | Internet of Things | Transportation

Components – EMEA

Engineering Solutions Center - Arrow EMEA

DragonBoard410c with Tresor Mezzanine

Arrow ESC
Revision 1.0, 23.11.2018

Quick Start Guide

Five Years Out

Revision History

Revision, Date	Editor	Subject (major changes)
Revision 1.0, 23.11.2018	ESC	First revision

Table of Contents

How to flash an Android Image on the DragonBoard

Hardware Requirements

The following equipment is needed to flash and run your DragonBoard™ 410c with Tresor:

- DragonBoard™ 410c + Power Supply
- Tresor Mezzanine Card
- Linux Host PC (Tested with Ubuntu)
- Micro-USB Cable (x2 flashing and debugging)

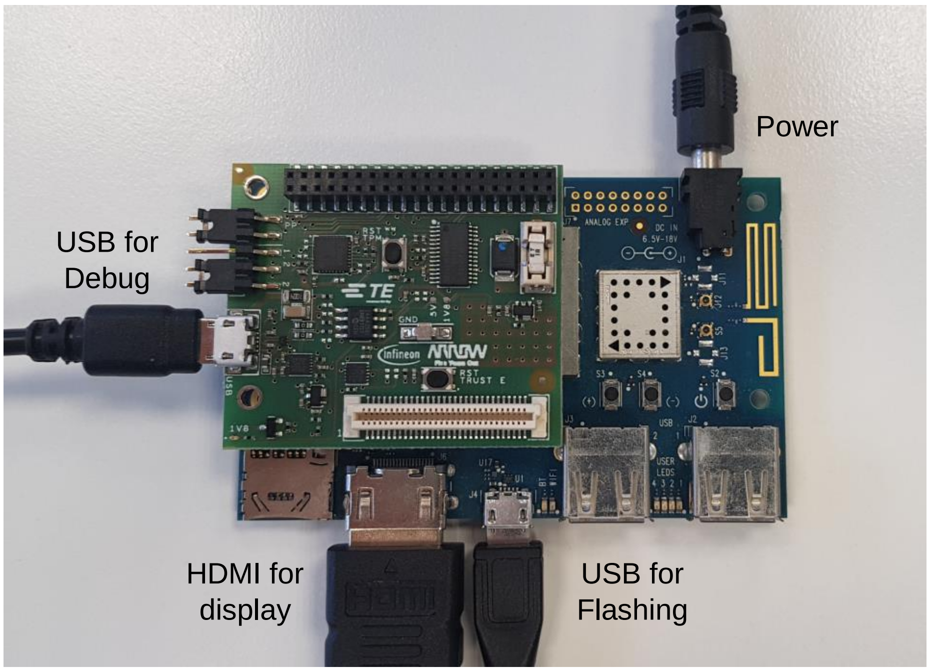

Figure 1 - Setup for flashing the DragonBoard™410c

Creating Modified Debian Image

Download latest prebuilt Debian developer (not installer) SD-card image from:

http://releases.linaro.org/96boards/dragonboard410c/linaro/debian/latest/

Write it to uSD card!

See the how-to under the following link:

https://www.96boards.org/documentation/consumer/dragonboard/dragonboard410c/installation/linux-sd.md.html

Check the “release notes” for preparing to rebuild the kernel on the linux host machine:

http://releases.linaro.org/96boards/dragonboard410c/linaro/debian/latest/

Start rebuilding the kernel with the following steps:

```
$ git clone -n http://git.linaro.org/landing-teams/working/qualcomm/kernel.git
$ cd kernel
$ git checkout -b kernel-18.01 debian-qcom-dragonboard410c-18.01
```

Modify the kernel by applying changes in kernel_diff.txt (available at arrow.com tresor product page):

```
$ git apply .../kernel_diff.txt
```

Continue and finish building the kernel following the remaining steps in the “release notes” up to building boot-db410c.img

HINT: on Ubuntu 16.04 one may use 'abootimg' instead of 'mkbootimg'.

Update the boot partition of the previously prepared uSD card (see “Release Notes”)

HINT: for updating without fastboot: one can simply write the Linux boot image to "boot" partition of the uSD card. To get partition names execute:

```
$ sudo parted /dev/sdX print -s
```

Do not forget to copy kernel modules to the 'rootfs' partition on the uSD card!

Attach Tresor v1.1 board to Dragonboard410c, set jumpers to enable TPM2.0 module, connect the USB-serial converter to PC, on the PC open a terminal emulator program and configure the serial port to 115200,N81 - see *Figure2*.

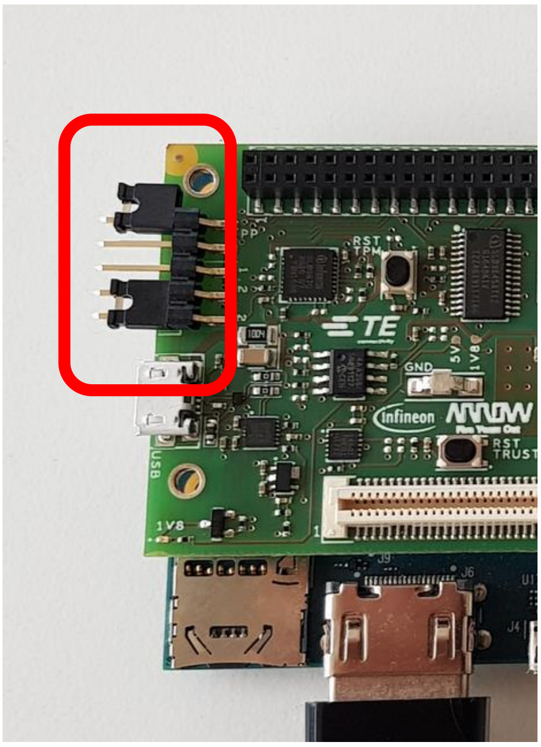

Figure 2 - Jumper set up

Insert uSD card, reboot and verify the presence of the TPM2 SPI driver with following commands:

```
# lsmod | grep tpm
          tpm_tis_spi             16384   0
          tpm_tis_core            20480   1 tpm_tis_spi

# ls /dev/ | grep tpm
          tpm0
          tpmrm0
```

Compiling and Testing tpm2-tss and tpm2-tss-engine

Configure WiFi as described here (don't download new SD-card image, use the running Linux):

https://discuss.96boards.org/t/headless-mode-with-wifi-vnc-server-and-xfce-desktop/5279

VNC is not needed, only WiFi.

Install the following packages:

```
$ su - linaro
$ sudo apt-get update
$ sudo apt-get -y install aptitude autoconf autoconf-archive automake \
  autotools-dev build-essential cmocka-doc libcurl4-gnutls-dev libgcrypt20-dev \
  libglib2.0 libglib2.0-dev libcmocka-dev libssl-dev libtool m4 man-db pandoc \
  pkg-config ruby-ronn uthash-dev zlib1g-dev
```

Let's get tpm2-tss and build it:

```
$ git clone https://github.com/tpm2-software/tpm2-tss
$ cd tpm2-tss/
$ ./bootstrap
$ ./configure --with-udevrulesdir=/etc/udev/rules.d --with-udevrulesprefix
$ make -j4
$ sudo make install
```

Same for tpm2-tools:

```
$ cd
$ git clone https://github.com/tpm2-software/tpm2-tools
$ cd tpm2-tools/
$ ./bootstrap
$ ./configure
$ make -j4
$ sudo make install
```

Same for tpm2-tss-engine:

```
$ cd
$ git clone https://github.com/tpm2-software/tpm2-tss-engine
$ cd tpm2-tss-engine/
$ ./bootstrap
$ ./configure
$ make -j4
$ sudo make install
$ sudo cp /usr/local/lib/openssl/engines/libtpm2tss.* /usr/lib/aarch64-linux-
gnu/engines-1.1/
$ sudo ldconfig
$ cd
$ sudo make install
```

Now we can verify tpm2-tools:

```
$ sudo tpm2_getcap -c ecc-curves
  TPM2_ECC_NIST_P256: 0x3
  TPM2_ECC_BN_P256: 0x10
```

Verify the OpenSSL TPM2 engine:

```
$ sudo openssl engine -t -c libtpm2tss
        (libtpm2tss) TPM2-TSS engine for OpenSSL
        Loaded: (tpm2tss) TPM2-TSS engine for OpenSSL
         [RSA, RAND]
         [ available ]
$ sudo openssl rand -engine libtpm2tss -hex 10 2>/dev/null
        ece9ade95f39d48f34f7
```

HINT: this last line of the code is a random HEX number generated by the Optiga TPM.

It is now possible to try all the openssl features mentioned here:

https://github.com/tpm2-software/tpm2-tss-engine/blob/master/README.md

Some caveats about the above page:

- always replace "-engine tpm2tss" with "-engine libtpm2tss" everywhere
- execute all the tools as root (eg. with 'sudo')
- sample calls sometimes have wrong arguments. Read the man pages of the given call for the correct arguments. For example:

```
'tpm2tss-genkey -a rsa -s 2048 mykey'      Instead of
'tpm2tss-genkey -a rsa -k 2048 mykey'
```

Here are some instructions about using commands of the tpm2-tools package to sign and verify documents:

https://dguerriblog.wordpress.com/2016/03/03/tpm2-0-and-openssl-on-linux-2/