



Android User Guide

REV. November 2018

Tetra (VL-EPC-2700)

ARM* i.MX6 Single Board Computer
with Gigabit Ethernet, Video, USB,
SATA, Serial I/O, Digital I/O, CAN
Bus, SPI, Mini PCIe, mSATA and I²C





WWW.VERSALOGIC.COM

12100 SW Tualatin Road
Tualatin, OR 97062-7341
(503) 747-2261
Fax (971) 224-4708

Copyright © 2018 VersaLogic Corp. All rights reserved.

Notice:

Although every effort has been made to ensure this document is error-free, VersaLogic makes no representations or warranties with respect to this product and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose.

VersaLogic reserves the right to revise this product and associated documentation at any time without obligation to notify anyone of such changes.

* Other names and brands may be claimed as the property of others.

Product Release Notes

Rev 1.0 First release for the Android Evaluation Kit

Support Page

The [Tetra Support Page](#) contains additional information and resources for this product including:

- Operating system information and software drivers
- Data sheets and manufacturers' links for chips used in this product
- BIOS information and upgrades

VersaTech KnowledgeBase

The VersaTech [KnowledgeBase](#) contains useful technical information about VersaLogic products, along with product advisories.

Customer Support

If you are unable to solve a problem after reading this manual, visiting the product support page, or searching the KnowledgeBase, contact VersaLogic Technical Support at (503) 747-2261. VersaLogic support engineers are also available via e-mail at Support@VersaLogic.com.

Repair Service

If your product requires service, you must obtain a Returned Material Authorization (RMA) number by calling 503-747-2261. Be ready to provide the following information:

Your name, the name of your company, your phone number, and e-mail address

The name of a technician or engineer that can be contacted if any questions arise

The quantity of items being returned

The model and serial number (barcode) of each item

A detailed description of the problem

Steps you have taken to resolve or recreate the problem

- The return shipping address
- Warranty Repair: All parts and labor charges are covered, including return shipping charges for UPS Ground delivery to United States addresses.

- **Non-warranty Repair:** All approved non-warranty repairs are subject to diagnosis and labor charges, parts charges and return shipping fees. Specify the shipping method you prefer and provide a purchase order number for invoicing the repair.

Note: Mark the RMA number clearly on the outside of the box before returning.

Contents

Introduction	6
Quick Start	7
EPC-2700 Tetra Hardware Introduction	7
Setting up Tetra for the First Time.....	8
Booting up Tetra for the First Time	8
Setting up the Development Host.....	11
Download the Android Source Code	13
Building the Android Image	14
Patching Source File.....	14
Setting up Build Configurations	14
Deploying the Image to Tetra	16
Updating U-boot	17
Advanced Tetra Features & Commands	19
CAN Network	19
References	21

Introduction

1

This document is intended to accompany the VersaLogic Android Starter Kit. It provides some basic information for setting up the hardware, but it mainly focuses on providing instructions to build and bring up the Android Operating System (OS) on the VersaLogic EPC-2700 Tetra Single Board Computer.

This Guide is not intended to provide significant amount of background information on the hardware included in the Kit or the Android OS. It is assumed that the user has the basic knowledge of Linux and the Android Operating System, and is able to obtain the necessary hardware and software that are required to complete the tasks outlined.

Note: The NXP ARM Cortex-A9 Quad processor was originally made by Freescale. Freescale was purchased by NXP, but much of the documentation still retains the name Freescale.

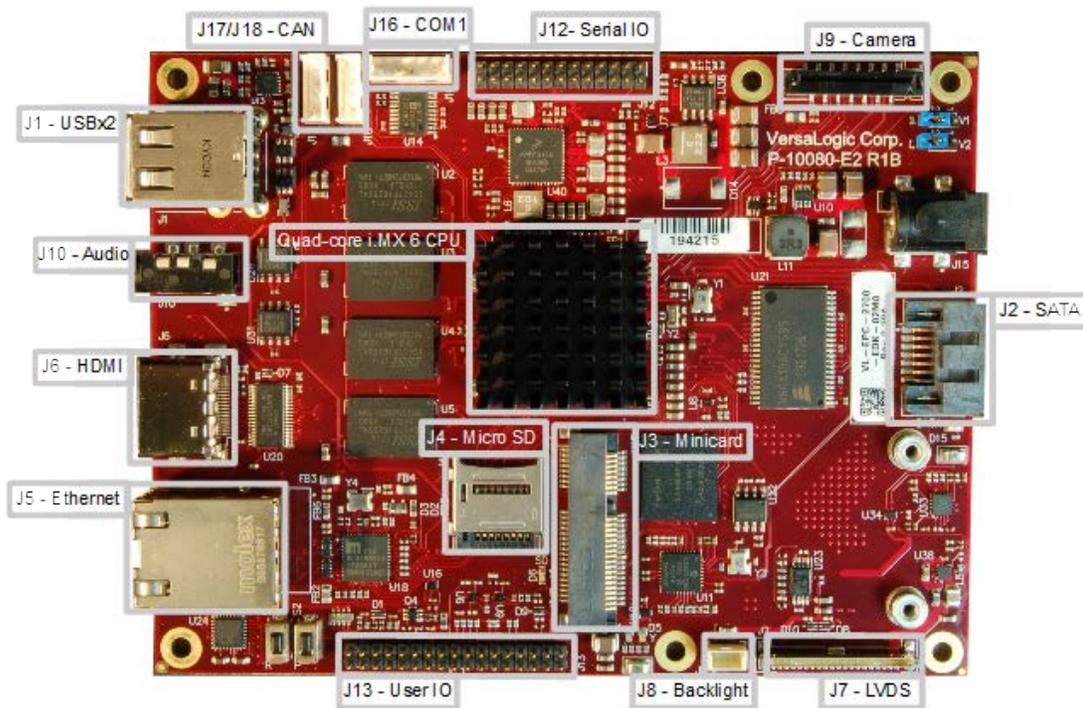
Quick Start

EPC-2700 Tetra Hardware Introduction

The VersaLogic EPC-2700 Tetra is an ARM based Single Board Computer. It consists of a NXP ARM Cortex-A9 Quad processor and many standard components. Detailed hardware description is available in the Tetra Hardware Reference Manual. The Tetra is included in the Android Starter Kit as the platform that will run the Android Operation System.

The following diagram shows the major components of a Tetra::

Figure 1. Major Components and Connectors



The Tetra will be shipped with power supply and a cable kit, which includes most of the required cables for the various ports shown above. A micro SD card with a demo image of the Android OS is also shipped with the cable kit.

Setting up Tetra for the First Time

At a minimum, these ports and devices should be connected before booting the Tetra for the first time:

- J1 – USBx2: There are two USB ports on the Tetra that support standard USB 2.0 devices. The bottom USB port is set to host mode, but the top USB OTG port is set to device mode by default. To change the top USB port to host mode, install V2 jumper per Tetra Hardware Reference Manual. To use the included touchscreen panel, its USB port needs to be connected to a powered USB hub first, and then connect the hub to the bottom USB port on the Tetra. If necessary, a USB mouse and keyboard can also be connected to the USB hub.
- J5 – Ethernet: connect network cable to the Ethernet port.
- J6 – HDMI: connect the touchscreen panel included in the Starter Kit to this port.
- J16 – COM1: connect the CRB-0504 adapter for RS-232 serial port, and then connect serial cable to a PC COM port. Use a terminal emulator such as PuTTY for console access to the Tetra.

Booting up Tetra for the First Time

At this point, the Tetra is ready to boot for the first time. Connect the power cable to the board and watch the serial console and you should see the boot messages displayed. Note that instead of BIOS, Tetra uses a different boot loader called U-Boot. Unless auto boot is stopped by pressing a key on the keyboard, U-Boot will load the Android OS.

Figure 2. Boot Messages

```

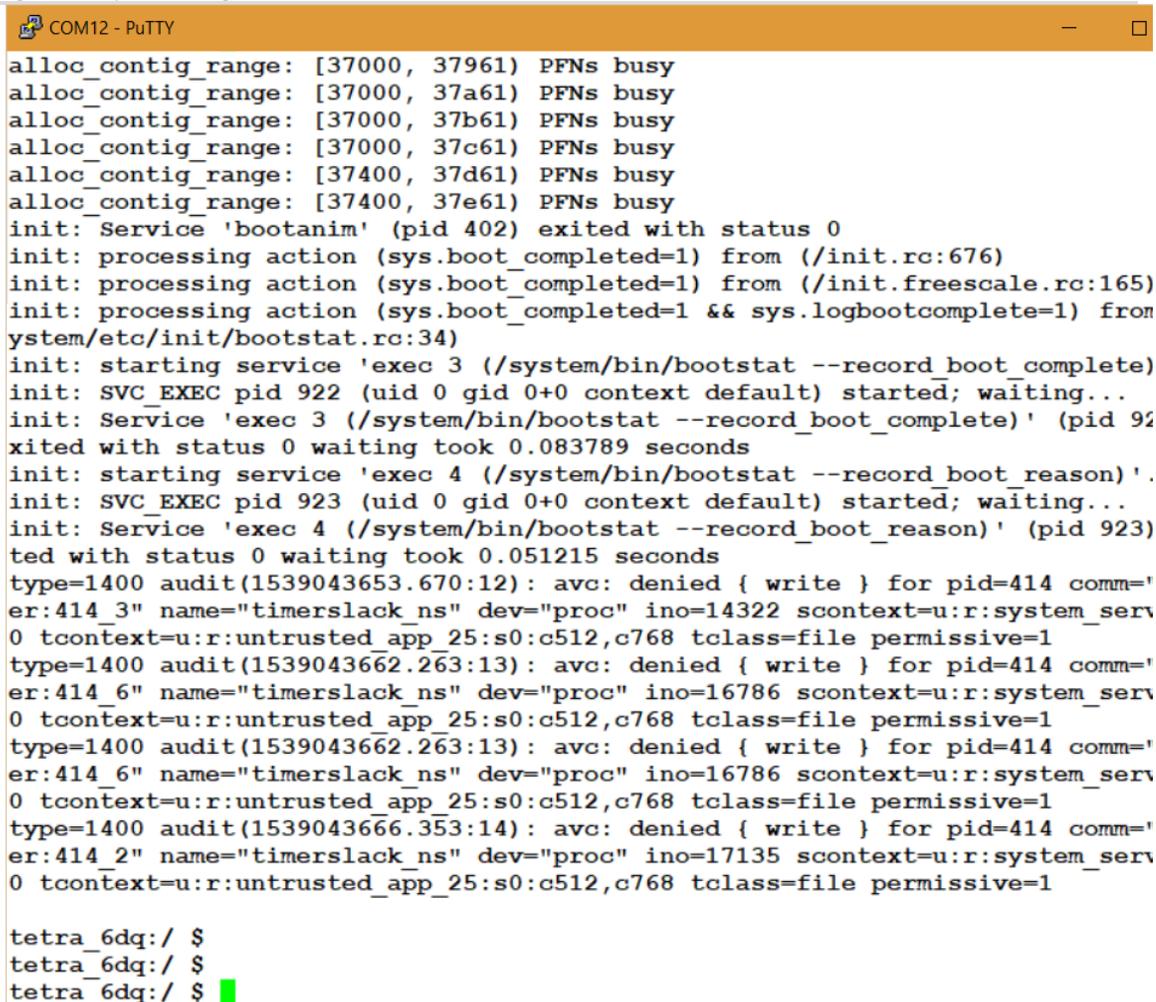
COM12 - PuTTY
Board: MX6-Tetra
DRAM: 2 GiB
MMC: FSL_SDHC: 0, FSL_SDHC: 1
SF: Detected w25q16cl with page size 256 Bytes, erase size 64 KiB, total 2 MiB
No panel detected: default to Hannstar-XGA
Display: Hannstar-XGA (1024x768)
In: serial
Out: serial
Err: serial
unsupported boot devices
Flash target is MMC:0
Net: No ethernet found.
Fastboot: Normal
Hit any key to stop autoboot: 0
boota mmc0
kernel @ 14008000 (9966136)
ramdisk @ 15000000 (2037914)
fdt @ 14f00000 (47772)
## Booting Android Image at 0x12000000 ...
Kernel load addr 0x14008000 size 9733 KiB
boot device type is incorrect.
Kernel command line: console=ttyMXC0,115200 androidboot.console=ttyMXC0 consoleb
lank=0 vmlloc=128M init=/init video=mxcfb0:dev=hdmi,1920x1080M@60,bpp=32 video=
mxcfb1:off video=mxcfb2:off video=mxcfb3:off androidboot.hardware=freescale cma=
448M androidboot.selinux=permissive androidboot.dmverity=disabled androidboot.s
torage_type=sd gpt no_console_suspend androidboot.serialno=130a19d4ea9a184b andr
oidboot.soc type=imx6q androidboot.soc type=imx6q
## Flattened Device Tree blob at 14f00000
Booting using the fdt blob at 0x14f00000
Loading Kernel Image ... OK
Using Device Tree in place at 14f00000, end 14f0ea9b

Starting kernel ...

```

Once the system boots, press the <Enter> key and a command prompt will be displayed. No log in is necessary.

Figure 3. System Login

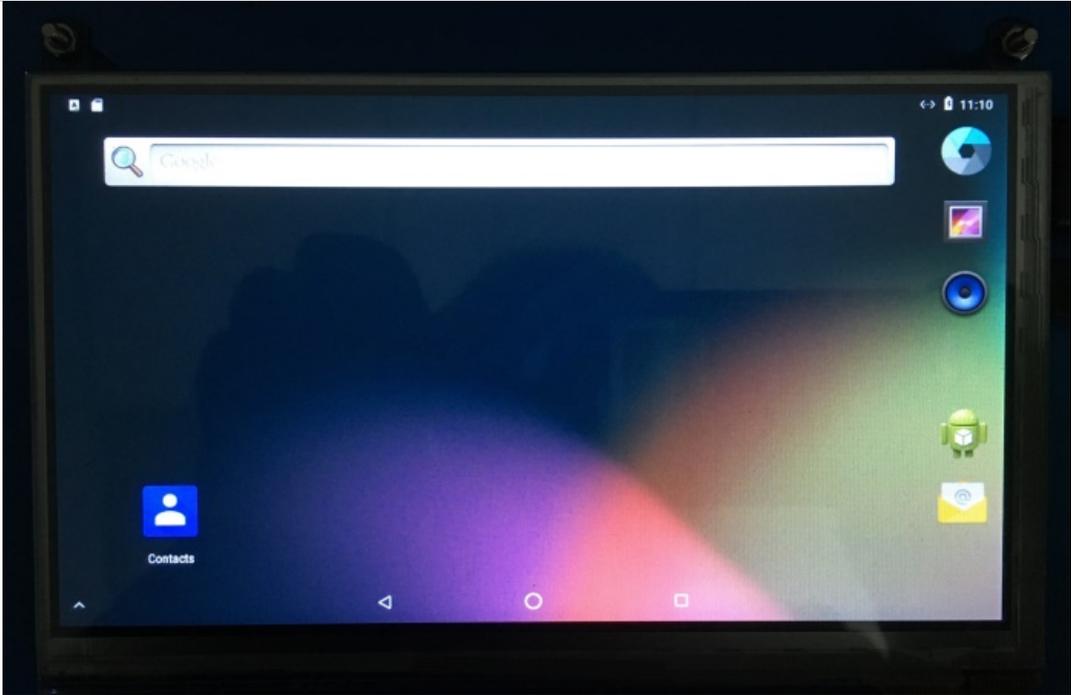


```
COM12 - PuTTY
alloc_contig_range: [37000, 37961) PFNs busy
alloc_contig_range: [37000, 37a61) PFNs busy
alloc_contig_range: [37000, 37b61) PFNs busy
alloc_contig_range: [37000, 37c61) PFNs busy
alloc_contig_range: [37400, 37d61) PFNs busy
alloc_contig_range: [37400, 37e61) PFNs busy
init: Service 'bootanim' (pid 402) exited with status 0
init: processing action (sys.boot_completed=1) from (/init.rc:676)
init: processing action (sys.boot_completed=1) from (/init.freescale.rc:165)
init: processing action (sys.boot_completed=1 && sys.logbootcomplete=1) from
system/etc/init/bootstat.rc:34)
init: starting service 'exec 3 (/system/bin/bootstat --record_boot_complete)
init: SVC_EXEC pid 922 (uid 0 gid 0+0 context default) started; waiting...
init: Service 'exec 3 (/system/bin/bootstat --record_boot_complete)' (pid 922)
exited with status 0 waiting took 0.083789 seconds
init: starting service 'exec 4 (/system/bin/bootstat --record_boot_reason)'
init: SVC_EXEC pid 923 (uid 0 gid 0+0 context default) started; waiting...
init: Service 'exec 4 (/system/bin/bootstat --record_boot_reason)' (pid 923)
ted with status 0 waiting took 0.051215 seconds
type=1400 audit(1539043653.670:12): avc: denied { write } for pid=414 comm='
er:414_3' name="timerslack_ns" dev="proc" ino=14322 scontext=u:r:system_serv
0 tcontext=u:r:untrusted_app_25:s0:c512,c768 tclass=file permissive=1
type=1400 audit(1539043662.263:13): avc: denied { write } for pid=414 comm='
er:414_6' name="timerslack_ns" dev="proc" ino=16786 scontext=u:r:system_serv
0 tcontext=u:r:untrusted_app_25:s0:c512,c768 tclass=file permissive=1
type=1400 audit(1539043662.263:13): avc: denied { write } for pid=414 comm='
er:414_6' name="timerslack_ns" dev="proc" ino=16786 scontext=u:r:system_serv
0 tcontext=u:r:untrusted_app_25:s0:c512,c768 tclass=file permissive=1
type=1400 audit(1539043666.353:14): avc: denied { write } for pid=414 comm='
er:414_2' name="timerslack_ns" dev="proc" ino=17135 scontext=u:r:system_serv
0 tcontext=u:r:untrusted_app_25:s0:c512,c768 tclass=file permissive=1

tetra_6dq:/ $
tetra_6dq:/ $
tetra_6dq:/ $ █
```

A standard Android screen as shown below will be displayed on the touchscreen panel.

Figure 4. The Android Screen



Setting up the Development Host

2

This chapter covers how to build and customize the Android OS running on Tetra.

A host PC or build server is required to setup the Android development environment. This device will be used to create the Operating System that will run on the VersaLogic Tetra board.

There are a few minimum requirements for this system:

1. Hardware – a server machine will provide the best performance, but at least a mid-range desktop PC is needed. Specifically, at least the following configuration:
 - a. 2 GHz dual core processor
 - b. 2 GB RAM (system memory)
 - c. 120 GB of free disk space is required in order to install the OS and required packages, and to build the target image. However, more disk space is highly recommended as multiple builds during typical development cycle can consume the disk space quickly.
 - d. VGA monitor capable of 1024x768 screen resolution
 - e. Either a CD/DVD drive or a USB port for the installer media
 - f. A static IP address is recommended but not required
 - g. Internet access to download additional required software
2. Operating System – the recommended OS for the host PC is Ubuntu 14.04, which is the version currently supported for Tetra that has been verified by VersaLogic. If the user decides to try a different version or Linux distribution, then it is up to the user to get the expected behavior on the host PC, as well as making sure the packages and utilities described below are compatible and can be installed correctly..

Host Packages

An Android build requires that some prerequisite software packages be installed. Please use the commands below to install these packages. Please also note that in Ubuntu, the command `sudo` is typically used to execute other commands with root privileges.

1. Install required software packages

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential  
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386  
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev ccache  
libgl1-mesa-dev libxml2-utils xsltproc unzip
```

```
$ sudo apt-get install android-tools-adb
```

```
$ sudo apt-get install uuid uuid-dev zlib1g-dev liblz-dev liblz02-2  
liblz02-dev lzop git-core curl u-boot-tools mtd-utils android-tools-  
fsutils device-tree-compiler gdisk
```

2. Install JDK

```
$ sudo add-apt-repository ppa:jonathonf/openjdk [or try ppa:openjdk-  
r/ppa]  
  
$ sudo apt-get update  
  
$ sudo apt-get install openjdk-8-jdk
```

Setting up the Repo Utility

Git is a version control system for tracking changes in source code files and coordinating work on those files among multiple developers. Repo is a tool built on top of Git that makes it easier to manage projects that contain multiple repositories, which may not be on the same server.

To install the “repo” utility, follow these steps:

1. Create a bin folder in the home directory.

```
$ mkdir ~/bin  
$ curl https://storage.googleapis.com/git-repo-downloads/repo >  
~/bin/repo  
$ sudo chmod a+x ~/bin/repo
```

2. Add the following line to the `.bashrc` file to ensure that the `~/bin` folder is in your `PATH` variable.

```
export PATH=~/bin:$PATH
```

3. Apply the new path to the current login session.

```
$ . ~/.bashrc
```

Download the Android Source Code

3

In order to download all the necessary files, first make sure that Git is setup properly with the commands below, which only need to be run once on the host PC.

```
$ git config --global user.name "Your Name"
(For example: git config --global user.name "John Doe")

$ git config --global user.email "Your Email"
(For example: git config --global user.email "John.Doe@acme.com")

$ git config --list
```

Download the source code from VersaLogic. As an example, a directory called `android_build` is created for this build, but any name can be used instead of this. By initializing and syncing the Git repo, VersaLogic source code is downloaded.

```
$ mkdir android_build
$ cd android_build

$ repo init -u git://github.com/versalogic/android-manifest.git -b
versalogic-imx-08.0.0_1.0.0-ga
$ repo sync
```

When this process completes, the source code is downloaded into the directory `android_build`. You can perform repo synchronization periodically with the command `repo sync` to update to the latest code. If errors occur during repo initialization, try deleting the `.repo` directory and running the repo initialization command again.

Since the Tetra utilizes the NXP ARM Cortex-A9 i.MX 6 Quad processor, the i.MX Android proprietary source code package `imx-o8.0.0_1.0.0_ga.tar.gz` also needs to be downloaded to the local build server. This file can be obtained from NXP at

Note: An NXP user account is required in order to download this file.

Once the file has been downloaded, unpack it to a temporary directory. Do not run the setup script `imx_android_setup.sh`, but instead manually copy the NXP vendor files to the build directory.

```
$ tar xvzf imx-o8.0.0_1.0.0_ga.tar.gz -C /tmp
$ cp -rfv /tmp/imx-o8.0.0_1.0.0_ga/vendor/nxp ~/android_build/vendor/
```

Building the Android Image

4

This section describes the necessary steps to building an Android image that will run on the VersaLogic Tetra board.

Patching Source File

There is an issue with the NXP source files. This can be fixed by executing the following command within the `android_build` directory:

```
$ cp prebuilts/ndk/r10/platforms/android-19/arch-arm/usr/include/linux/android_pmem.h device/fsl/common/kernel-headers/linux
```

Setting up Build Configurations

First set up the build environment. Note this only configures the current terminal.

```
$ source build/envsetup.sh
```

The command **lunch <buildName-buildType>** is used to set up the build configuration. For the Tetra board, the **buildName** is "tetra_6dq". There are 3 build types as shown below:

Build Type	Description
user	Production ready image, no debug
userdebug	Production ready image similar to "user" but with root access and debug tools. Be sure to use this option if serial console connection via J16-Com1 port is desired.
eng	Development image with debug tools

To build a production ready image with console support, execute the following command:

```
$ lunch tetra_6dq-userdebug
```

Creating the Target Image

Finally, use the `make` command to compile all the source code and create the Android image. This process can take several hours to complete.

```
$ make 2>&1 | tee build-log.txt
```

Once the build process successfully completes, the Android image files will be created in the `out/target/product/tetra_6dq` directory. Some of the relevant files are listed below:

- `root/`: root file system (including `init`, `init.rc`). Mounted at `/`.
- `system/`: Android system binary/libraries. Mounted at `/system`.
- `data/`: Android data area. Mounted at `/data`.
- `recovery/`: root file system when booting in "recovery" mode. Not used directly.
- `boot-imx6q.img`: composite image for i.MX 6Dual/6Quad, which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
- `ramdisk.img`: ramdisk image generated from "root/". Not used directly.
- `system.img`: EXT4 image generated from "system/". It can be programmed to "SYSTEM" partition on SD/eMMC card with "dd".
- `recovery-imx6q.img`: EXT4 image for i.MX 6Dual/6Quad, which is generated from "recovery/". Can be programmed to the "RECOVERY" partition on SD/eMMC card with "dd".
- `partition-table.img`: GPT partition table image, used for 8 GB SD card.
- `partition-table-14GB.img`: GPT partition table image, used for 16 GB SD card.
- `partition-table-28GB.img`: GPT partition table image, used for 32 GB SD card
- `u-boot-imx6q.imx`: U-Boot image with no padding for i.MX 6Dual/6Quad.
- `vendor.img`: vendor image, which holds platform binaries, mounted at `/vendor`.

5

Deploying the Image to Tetra

The simplest method to deploy the image is by using a MicroSD card. Use an internal or external SD card reader to attach the target MicroSD card to the host PC. Be sure to unmount all the SD card partitions. Flash the image onto the card with the following commands:

```
$ cd ~/android_build/out/target/product/tetra_6dq

$ sudo ~/android_build/device/fsl/common/tools/fsl-sdcard-partition.sh
-f I mx6q [-c 14/28] /dev/sdX
```

Whereas the optional parameter “-c” is not required if using an 8 GB MicroSD card. “-c 14” is required to use a 16 GB card and “-c 28” required for 32 GB card. /dev/sdX denotes the device name of the MicroSD card.

Note that the script “fsl-sdcard-partition.sh” requires the `simg2img` tool to be installed on the host PC. `simg2img` is a tool that converts the sparse system image to raw system image on the host PC. The `android-tools-fsutils` package includes the `simg2img` command for Ubuntu Linux OS. Additionally, `sfdisk` v2.26 or later is also required to run this script.

After the image has been successfully flashed onto the MicroSD card, the card can be inserted into the Tetra board to boot it up, as described in the Quick Start section of this guide.

Updating U-boot

6

As part of the Android Starter Kit, VersaLogic has customized U-boot to specifically support Android on the Tetra board. During the manufacturing process, this customized version is burned onto the SPI flash, which is the default U-boot device. The user can further customize U-boot behavior using the environment variables described below. If the user needs any particular customization beyond these, then please contact VersaLogic for support.

The U-boot environment variables are also preprogrammed to work on the Tetra. However, if any change is required, then please follow these instructions:

1. Power on the Tetra board and stop auto boot by pressing any key, and then use the `printenv` command to list all variables that are currently defined:

Figure 5. The Defined Variables List

```

COM12 - PuTTY
DRAM:  2 GiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
SF: Detected w25q16cl with page size 256 Bytes, erase size 64 KiB, total 2 MiB
No panel detected: default to Hannstar-XGA
Display: Hannstar-XGA (1024x768)
In:    serial
Out:   serial
Err:   serial
unsupported boot devices
flash target is MMC:0
Net:   No ethernet found.
Fastboot: Normal
Hit any key to stop autoboot:  0
=> printenv
baudrate=115200
board_name=TETRA
board_rev=MX6Q
bootargs=console=ttymx0,115200 androidboot.console=ttymx0 consoleblank=0 vmlinux=128M init=/init video=mx0fb0:dev=hdmi,1920x1080M@60,bpp=32 video=mx0fb1:off video=mx0fb2:off video=mx0fb3:off androidboot.hardware=freescale cma=448M androidboot.selinux=permissive androidboot.dm_verity=disabled androidboot.storage_type=sd gpt no_console_suspend
bootcmd=boota mmc0
bootdelay=3
ethprime=eth0
fastboot dev=mmc0
fdt_high=0xffffffff
fdtControladdr=8ef30538
initrd_high=0xffffffff
loadaddr=0x12000000
soc_type=imx6q
splashpos=m,m

Environment size: 602/8188 bytes
=>

```

2. If any variable needs to be changed, then select the appropriate command from the examples below:

```
=> setenv fastboot_dev mmc0
=> setenv bootcmd boota mmc0
=> setenv bootargs console=ttymx0,115200 androidboot.console=ttymx0
consoleblank=0 vmalloc=128M init=/init
video=mxcfb0:dev=hdmi,1920x1080M@60,bpp=32 video=mxcfb1:off
video=mxcfb2:off video=mxcfb3:off androidboot.hardware=freescale
cma=448M androidboot.selinux=permissive androidboot.dm_verity=disabled
=> saveenv
=> reset
```

In case a newer version of U-boot is available, the Tetra can be updated following these steps:

1. Copy the new U-boot file to a TFTP server. For example, copy the file `u-boot-imx6q.imx` to `/tftpboot` on the host PC.
2. Power on Tetra and stop U-boot auto boot. Execute the following commands to update:

```
=> dhcp
=> setenv serverip <Host PC IP>
=> tftp 0x10800000 <U-boot file name>
=> sf probe
=> sf erase 0 0xc0000
=> sf write 10800000 0x400 $filesize
```

3. Reboot the Tetra and verify that the U-boot version number has been updated.

Note that currently there is an issue with Android U-boot, so it does not support networking. Therefore the above instructions do not apply. This issue should be addressed in the future. Please contact VersaLogic for support if upgrading the Android U-boot is necessary.

Advanced Tetra Features & Commands



This chapter describes some of the advanced features available on the Tetra running the Android OS. More information may be added in the future as it become available.

CAN Network

There are two CAN ports on the Tetra, which can be used to connect to other CAN capable devices. The CAN network device driver interface provides a generic interface to setup, configure and monitor CAN network devices.

1. To list the CAN interfaces, first become root and then execute the `ifconfig` command in a terminal window:

```
tetra_6dq:/ # su
tetra_6dq:/ # ifconfig -a
...
can1      Link encap:UNSPEC      Driver flexcan
          NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 TX bytes:0
          Interrupt:36

can0      Link encap:UNSPEC      Driver flexcan
          NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 TX bytes:0
          Interrupt:35
```

The user is able to configure the CAN interfaces, like setting the bit-timing parameters, using the command "ip." A CAN network device is started or stopped with the command "ifconfig canX up/down" or "ip link set canX up/down." Be sure to define proper bit timing parameters for real CAN devices before starting it to avoid error-prone default settings. The following example is a test scenario which illustrates the use of these commands.

1. Connect the two CAN ports with a cable to run a loopback test.
2. Execute the following commands in a terminal to configure the can0 and can1 interfaces:

```
sh-4.3# ip link set can0 up type can bitrate 125000
sh-4.3# ip link set can1 up type can bitrate 125000
```

3. `ifconfig` command should now show that both CAN interfaces are up:

```
sh-4.3# ifconfig -a
```

```
...
can1    Link encap:UNSPEC    Driver flexcan
        UP RUNNING NOARP  MTU:16  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:10
        RX bytes:0 TX bytes:0
        Interrupt:36

can0    Link encap:UNSPEC    Driver flexcan
        UP RUNNING NOARP  MTU:16  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:10
        RX bytes:0 TX bytes:0
        Interrupt:35
```

References

A red square containing a white number '8' in a bold, sans-serif font.

Information on the NXP ARM Cortex-A9 Quad processor is available at <https://www.nxp.com/products/microcontrollers-and-processors/arm-based-processors-and-mcus/i.mx-applications-processors/i.mx-6-processors/i.mx-6quad-processors-high-performance-3d-graphics-hd-video-arm-cortex-a9-core:i.MX6Q>

NXP support page for Android OS for i.MX Applications Processors: <https://www.nxp.com/support/developer-resources/run-time-software/i.mx-developer-resources/android-os-for-i.mx-applications-processors:IMXANDROID>

Ubuntu 14.04 can be downloaded from <https://www.ubuntu.com/download/alternative-downloads>