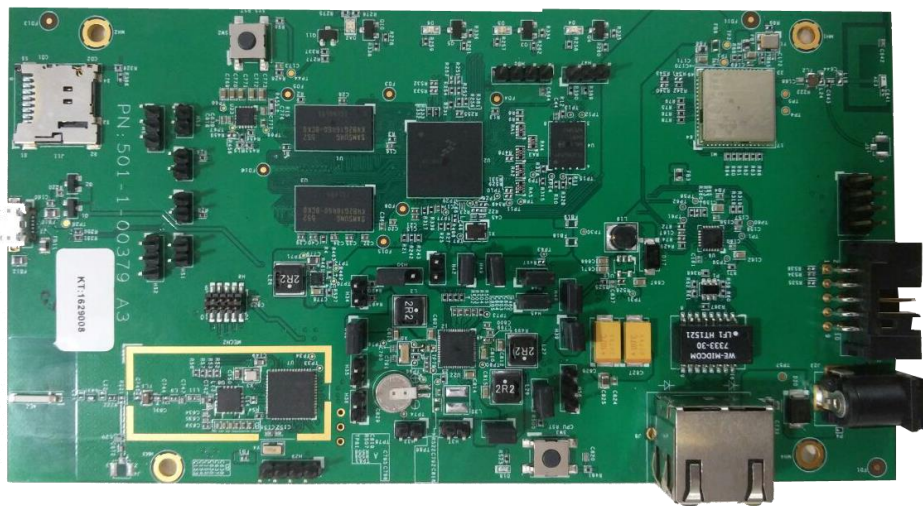


TPS65910 IoT Gateway Reference Design Using i.MX6 SL User Guide



Revision 1.0

Copyright Notice

This document is copyrighted material of RadiumBoards, All Rights Reserved. No part of this document, in whole or in part, may be used, reproduced, stored in a retrieval system or transmitted, in any form, or by any means, electronic or otherwise, including photocopying, reprinting, or recording, for any purpose, without the express written permission of RadiumBoards.

Legal Disclaimer

The information contained in this document is subject to change without notice. The information in this document is provided for informational purposes only. RadiumBoards specifically disclaims all warranties, express or limited, including, but not limited, to the implied warranties of merchantability and fitness for a particular purpose, except as provided for in a separate software license agreement.

RadiumBoards

RadiumBoards is a unique website providing complete board and software solutions addressing a broad range of markets including Security and Surveillance, Networking, Wireless, Video, Audio, Automotive, Mobile Device and IOT (Internet of Things).

OEMs and Systems Integrators can shop for complete assemblies, with firmware, BSP and applications ready to integrate into your own enclosures.

ODMs can shop for reference designs complete with full BSP (Board Support Package) and application support. Radium Boards can also provide any level of customization to the hardware or software required by the ODM to help differentiate in their markets.

Electronics hardware and software hobbyists, experimenters and educators now have access to complete high performance platforms for application in an infinite range of projects.

Correspondence

Corporate Office

B-22, Infocity Sector-34,
Gurgaon-122001, Haryana, India

Tel No: +91 124 4284250

US Office

2025 Gateway Place, Suite 465
San Jose, California 95110

E-MAIL

info@radiumboards.com

WEBSITE

www.radiumboards.com

VVDN Technologies

VVDN Technologies Pvt. Ltd. is a sibling company of RadiumBoards and is responsible for the design and development of all products sold through the RadiumBoards brand.

Founded in 2007, VVDN is a technology innovation and development company providing a broad spectrum of services and technology expertise to our core domains. VVDN provides “Concept to Customer” services at any point in the development cycle, as well as full turnkey solutions.

WEBSITE

www.vvdntech.com

Table of Contents

1.	ABOUT THIS DOCUMENT	6
1.1	Document Conventions.....	6
1.2	Terms and Abbreviations	6
2.	INTRODUCTION.....	7
2.1	Product Overview.....	7
2.2	Software Scope of the Project	8
2.3	Software Design and Development	8
3.	SOFTWARE ARCHITECTURE	9
4.	BUILD SETUP AND GUIDE.....	10
4.1	Yocto Project community layer	10
4.2	Host Setup.....	10
4.3	Host Packages.....	10
4.4	Yocto Project Setup.....	11
4.5	Image Build.....	12
4.6	Machine Configurations.....	12
4.7	Bitbake options.....	13
4.8	Building an Image	14
4.9	Image Deployment	14
4.10	Flashing the SD Card Image	14
5.	BOARD BRING-UP [FIRST BOOT MECHANISM]	15
5.1	USB boot mode.....	15
1.1.1	Pre-requirements	15
1.1.2	Board detection in PC	15
1.1.3	Loading u-boot image	16
5.2	SPI Boot Mode	16
1.1.4	Writing images into flash	16
1.1.5	Setting environment variables	17
5.3	SD Card boot mode.....	17
4.3.1	Format the SD card.....	17
6.	LEDS NOTIFICATIONS	20
7.	WI-FI INTERFACE	20
7.1	Client mode	20
6.1	AP mode.....	22
6.2	WLAN to Ethernet Bridging	23
6.3	Wi-Fi Throughput	24
8.	BLUETOOTH INTERFACE.....	25
9.	ZIGBEE INTERFACE	26
9.1	ZigBee to WLAN Bridging	27
10.	ETHERNET INTERFACE.....	30
10.1	Interface test.....	30
10.2	Throughput	30
11.	SD CARD INTERFACE.....	31
12.	TPS65910 PMIC INTERFACE	32
12.1	Low-level Power Management (PM)	32
12.2	Software Operation	32
12.3	Wakeup from low-power modes	33
12.4	Dynamic Voltage Frequency Scaling Operation	33

1.1.6	Software Operations.....	35
13.	BOARD BUTTON FUNCTIONALITIES	36
14.	DHRYSTONE BENCHMARK	36
14.1	Dhrystone Performance Calculation	36
15.	RTC	38

Figures

FIGURE 1	SYSTEM BLOCK DIAGRAM	8
FIGURE 2	SOFTWARE ARCHITECTURE DIAGRAM ON I.MX6 SL.....	9
FIGURE 3	DVFS ARCHITECTURE	34

Tables

TABLE 1:	DOCUMENT CONVENTIONS	6
TABLE 2:	TERMS AND ABBREVIATIONS.....	6

1. About This Document

This document provides details of the TPS65910 Based IOT Gateway including its features, functionality, installation and configuration.

1.1 Document Conventions

The different conventions used in this document are explained in the following table:




Convention	Description
	Note: Provides information about important features or instructions.
	Caution: Alerts you to potential damage to a program, device, or system.
	Warning: Alerts you to potential injury or fatality and to potential electrical hazards.
Bold font	Any option that needs to be selected or typed in the user interface is represented using bold font.

Table 1: Document Conventions

1.2 Terms and Abbreviations

The different terms and abbreviations used in this document are explained below

Table 2: Terms and Abbreviations

Terms / Abbreviation	Description / Expansion
ADAS	Advanced Driving Assistance System
VS	Video Security
DSS	Display Subsystem
GUI	Graphical User Interface
LSP	Linux Support Package
MB/s	Mega Byte per second
Mbps	Mega bit per second
OSD	On Screen Display
PTZ	Pan Tilt Zoom
TOF	Time of Flight
RTSP	Real Time Streaming Protocol
SDK	Software Development Kit
UI	User Interface
MSFP	Multi Sensor Fusion Platform

2. Introduction

This document describes the software design of IOT gateway to be designed for Texas Instrument. These requirements have been derived from the requirement specifications provided by the customer, VVDN's business proposal to customer and subsequent discussions with the customer and agreed upon product requirement document.

This SDD is made for the reference of

- Product managers at VVDN/TI to confirm the architecture before/during development.
- Engineering team at VVDN for System Architecture, Design and development of TlxU_PMIC.
- System Integration and Verification team at VVDN for firmware development and validation plan drafting.

2.1 Product Overview

IOT gateway functions as a gateway between a ZigBee network and an IP network through Ethernet and Wi-Fi. On the ZigBee network side, it acts as a node talking to a ZigBee enable smart devices like light nodes and sensors.

On the IP network side, IOTG streams the collected information to smart devices like phones and provides remote control option from these devices as well.

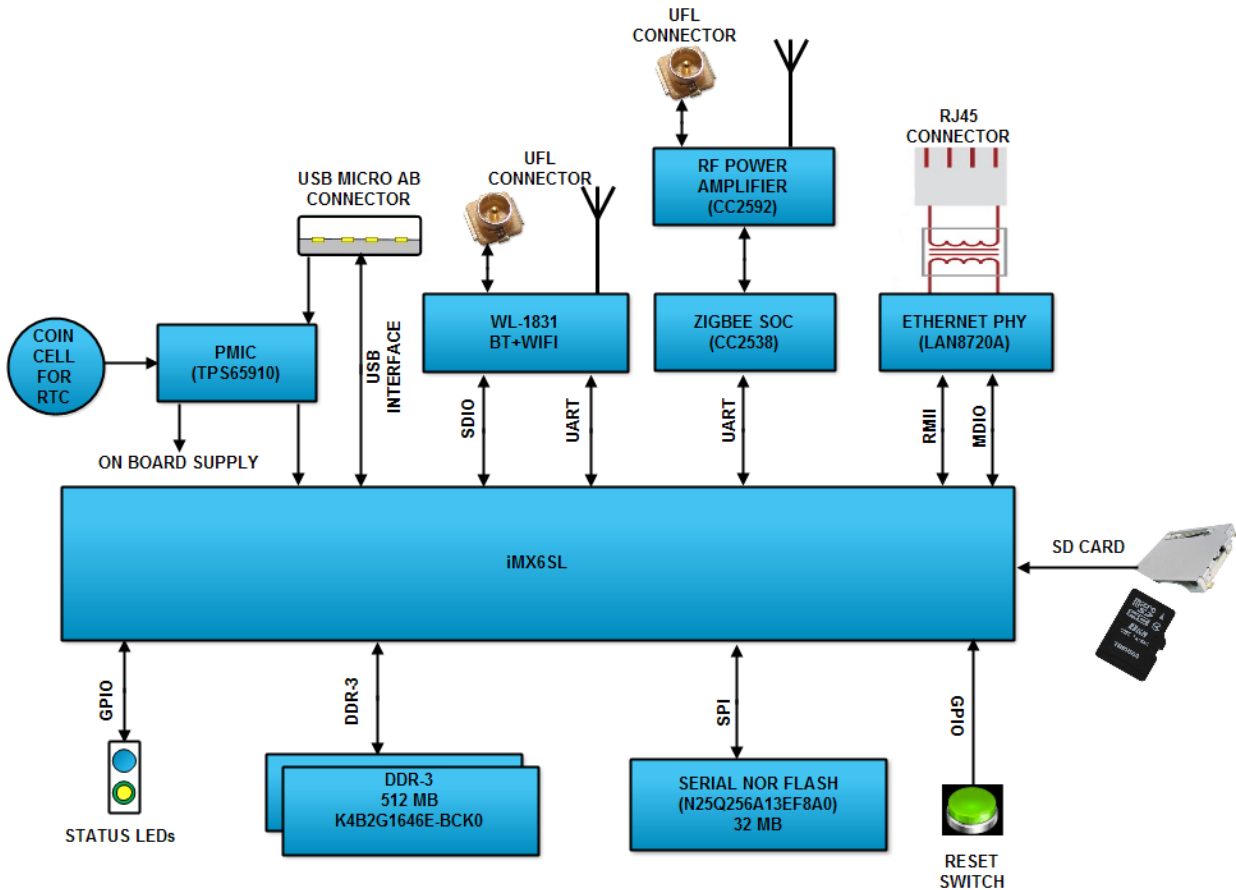


Figure 1 System Block diagram

2.2 Software Scope of the Project

The main scope of the project is to design, develop and deliver the hardware and software for IOT gateway meeting all the requirements specified in the PRD.

2.3 Software Design and Development

- Software Design and Development
- Boot loader and Linux Porting
- Linux device driver porting/development for all the peripherals
- The board will support the following low power modes
- RUN, WAIT, STOP, DORMANT which are maps to kernel power management systems like standby, Mem (suspend to RAM), freeze (lo-power idle).
- Board Bring-up
- Testing and Validation

Following documentations will be provided

1. **User Guide:** - Illustrating how to boot and flash the board.
2. **Software Developer's Guide:-** How to rebuild all software provided
3. **Test Report:-** What use cases will be tested and the equipment used for the testing
4. **Power consumption document:-** Maps to Freescale document showing power consumption for various use cases

3. Software Architecture

The software architecture diagram on i.MX6 SL is shown below:

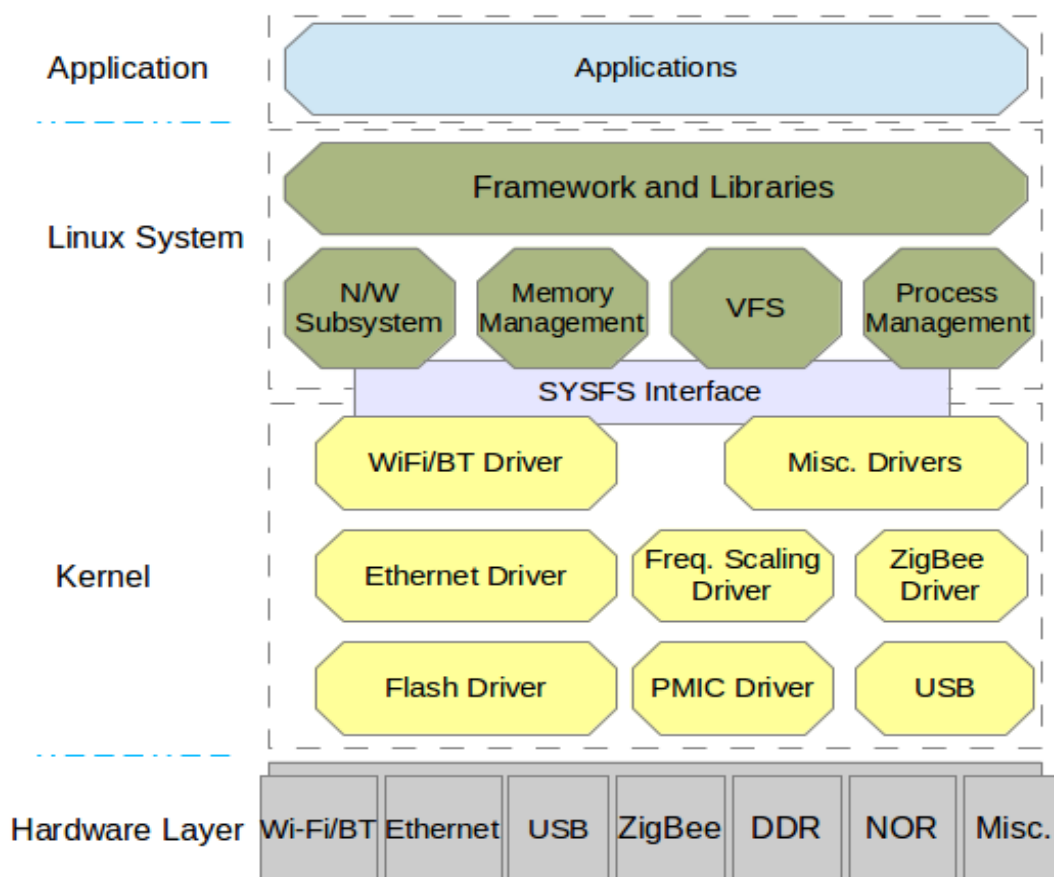


Figure 2 Software architecture diagram on i.MX6 SL

4. BUILD setup and guide

The Yocto Project is an open-source collaboration focused on embedded Linux OS development.

Files used to build an image are stored in layers. Layers contain different types of customizations and come from different sources. Some of the files in a layer are called recipes. Yocto Project recipes contain the mechanism to retrieve source code, build and package a component.

4.1 Yocto Project community layer

Following are the layers which are used in this project:

meta-fsl-arm: provides support for the base and for Freescale ARM reference boards

meta-fsl-arm-extra: provides support for 3rd party and partner boards

meta-fsl-demos: additional items to aid in development

base: Provides base configuration for FSL Community BSP

meta-openembedded: Collection of layers for the OE-core universe. See layers.openembedded.org/.

poky: basic Yocto Project items in Poky. See the Poky README for details

meta-vvdn: customized layer for the configuration of kernel and u-boot.

4.2 Host Setup

To get the Yocto Project in a Linux Host Machine, the packages and utilities described below must be installed. When building on a machine running Ubuntu, the minimum hard disk space required is about 50 GB.

The recommended minimum Ubuntu version is 12.04 or later. Earlier versions may cause the Yocto Project build setup to fail, because it requires python versions only available starting with Ubuntu 12.04.

4.3 Host Packages

Essential Yocto Project host packages:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
```

```
build-essential chrpath socat
```

i.MX layers host packages for a **Ubuntu 12.04 or 14.04** host setup:

```
$ sudo apt-get install libstdl1.2-dev xterm sed cvs subversion coreutils texi2html docbook-utils  
python-pysqlite2 help2man make gcc g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev  
mercurial autoconf automake groff curl lzop asciidoc
```

i.MX layers host packages for a **Ubuntu 12.04** host setup only:

```
$ sudo apt-get install uboot-mkimage
```

i.MX layers host packages for a Ubuntu 14.04 host setup only:

```
$ sudo apt-get install u-boot-tools
```

4.4 Yocto Project Setup

The Freescale Yocto Project BSP Release directory contains a "sources" directory, which contains the recipes used to build, one or more build directories, and a set of scripts used to set up the environment.

The recipes used to build the project come from both the community and Freescale. The Yocto Project layers are downloaded and placed in the sources directory. This sets up the recipes that are used to build the project.

Fetch complete Yocto repository and build for tixu board

```
$ git clone https://github.com/sanojvvdn/tixu-src.git
```

Fetch Standalone Kernel:

```
$ git clone https://github.com/sanojvvdn/linux-kernel-3.14.git
```

Fetch Standalone u-Boot:

```
$ git clone https://github.com/sanojvvdn/u-boot-2015.git
```

```
$ cd tixu-src/
```

Then after set the machine configuration

```
$ export MACHINE=imx6sltixu
```

Setup the environment for building the image

```
$ source ./setup-environment build
```

Build the image

```
$ bitbake core-image-minimal
```

4.5 Image Build

This section provides the detailed information along with the process for building an image.

Image Name	Target	Provided by layer
core-image-minimal	A small image that only allows a device to boot.	poky
core-image-base	A console-only image that fully supports the target device hardware.	poky
core-image-sato	An image with Sato, a mobile environment and visual style for mobile devices. The image supports X11 with a Sato theme, Pimlico applications. It contains a terminal, an editor and a file manager.	poky
fsl-image-machine-test	An FSL Community i.MX core image with console environment - no GUI interface	poky
fsl-image-gui	Builds a Freescale image with a GUI without any QT content. This image recipe works on all backends for X11, DirectFB, Frame Buffer and Wayland	meta-fsl-bsp-release/imx/meta-fsl-demos
fsl-image-qt5	Builds a QT5 image for X11, Frame Buffer and Wayland backends	meta-fsl-bsp-release/imx/meta-fsl-demos

4.6 Machine Configurations

Freescale provides new or updated machine configurations that overlay the meta-fsl-arm machine configurations. These files are copied into the meta-fsl-arm/conf/ machine directory.

The following are all the Freescale machine configuration files that can be selected:

- imx6dlsabreauto
- imx6dlsabresd
- imx6qsabreauto
- imx6qsabresd
- imx6slevk
- imx6sltixu
- imx6solosabreauto
- imx6solosabresd
- imx6sxsabresd

- imx6sxsabreauto

To see all MACHINE options:

```
$ source setup-environment
```

If undefined, this script will set \$MACHINE to 'imx6sltixu'.

The EULA must be accepted the first time. After that, the acceptance is logged and EULA acceptance is not required again.

The i.MX machine files are provided in meta-fsl-arm/conf/machine

The MACHINE configuration can also be changed in <builddir>/conf/local.conf .

The following is a part of a local.conf created from the setup-environment script:

```
MACHINE ??= 'imxsl6tixu'
```

```
DISTRO ?= 'poky'
```

```
ACCEPT_FSL_EULA = "1"
```

4.7 Bitbake options

The bitbake command used to build an image is bitbake <image name>. Additional parameters can be used for specific activities described below. Bitbake provides various useful options for developing a single component. To run with a bitbake parameter, the command looks like this:

```
bitbake <parameter> <component>
```

<Component> is a desired build package.

The following table provides some bitbake options.

Bitbake paramater	Description
-c fetch	Fetches if the downloads state is not marked as done.
-c cleanall	Cleans the entire component build directory. All the changes in the build directory will be lost. The rootfs and state of the component are also cleared. The component is also removed from the download directory.
-c deploy	Deploys an image or component to the rootfs.

-k	Continues building components even if a build break occurs.
-c compile -f	It is not recommended that the source code under the tmp directory is changed directly, but if it is, the Yocto Project might not rebuild it unless this option is used. Use this option to force a recompile after the image is deployed.
-g	Lists a dependency tree for an image or component.
-DDD	Turns on debug 3 levels deep. Each D adds another level of debug.

4.8 Building an Image

The Yocto Project build uses the bitbake command. For example, bitbake <component> builds the named component.

Each component build has multiple tasks, such as fetching, configuration, compilation, packaging, and deploying to the target rootfs. The bitbake image build gathers all the components required by the image and build in order of the dependency per task. The first build is the toolchain along with the tools required for the components to build.

The following command is an example on how to build an image:

```
$ bitbake core-image-minimal
```

4.9 Image Deployment

After a build is complete, the created image resides in <build directory>/tmp/deploy/images. An image is, for the most part, specific to the machine set in the environment setup. Each image build creates a U-Boot, a kernel, and an image type based on the IMAGE_FSTYPES defined in the machine configuration file. Most machine configurations provide an SD card image (.sdcard), an ext3 and tar.bz2. The ext3 is the root file system only. The .sdcard image contains U-Boot, the kernel and the rootfs completely set up for use on an SD card.

4.10 Flashing the SD Card Image

An SD card image provides the full system to boot with U-Boot and kernel. To flash an SD card image, run the following command:

```
$ sudo dd if=<image name>.sdcard of=/dev/sd<partition> bs=1M && sync
```

5. Board bring-up [First Boot mechanism]

The board can be booted in 3 modes

1. USB mode [Header H1 & H2 Connected]
2. SPI boot mode [No Header Connected]
3. SD card boot mode (Default booting media) [Sort Pins 1&2, 3&4 of H31]

By default the board will be boot in SD Card mode.

Four images needs to be programmed to the board:

Images	Descriptions
u-boot.imx	Boot loader image
ulmage	Linux Kernel Image
uramdisk.image.gz	File system image
imx6sl-tixu.dtb	Device tree image

5.1 USB boot mode

This mode is used to boot the board from USB. User will be able to boot up to boot-loader (u-boot) using this mode. This boot mode is intended for first time programming of the board.

1.1.1 Pre-requirements

- **Micro USB Cable:** This is used for powering the board initially and loading image for USB boot mode.
- **A Linux PC:** 'imx_usb' tool needs to be executed on linux PC for booting. (Tested in Ubuntu 12.04 & Ubuntu 14.04LTS)
- **Ethernet Cable:** For using TFTP in order to program linux images from u-boot console. Default IP address of board is 192.168.1.200.
- **UART Console Cable:** Serial console cable for accessing the board console in PC (through terminal). Baud rate is 115200.

1.1.2 Board detection in PC

If the board is connected to the PC in USB boot mode it will list as below in the output of lsusb command.

"Freescale Semiconductor, Inc"

1.1.3 Loading u-boot image

An image for u-boot is loaded into RAM through USB cable.

1. Open two terminals. In one terminal open imx_usb_loader tool and in another terminal open minicom for serial console of board.
2. To load u-boot image, following command is used. This command should be executed in imx_usb_loader tool folder of x86 machine. (It is available in tools directory in the source code).

Get the tool here:

[git clone git://github.com/boundarydevices/imx_usb_loader](https://github.com/boundarydevices/imx_usb_loader)

Compile it on x86 machine:

```
$ cd imx_usb_loader
```

```
$ make
```

```
$ sudo ./imx_usb<path_for_u-boot_image>
```

1. To stop at u-boot prompt press any key on console window opened using minicom.

5.2 SPI Boot Mode

1.1.4 Writing images into flash

Load the u-boot image using USB boot mode.

TFTP server is needed to load the kernel image and rootfs in RAM.

(For tftp setup follow the below link)

<https://mohammadthalif.wordpress.com/2010/03/05/installing-and-testing-tftpd-in-ubuntudebian/>.

This is only for example. User can follow any tftp installation method. To write images into flash memory following commands are used.

Programming to NOR Flash :

```
=====
Writing u-boot image
=====
tftp 0x80800000 u-boot.imx
sf probe
sf erase 0x0 0x60000
sf write 0x80800000 0x400 0x60000
```

After writing u-boot image, reboot the board by power off/on. Now u-boot automatically will be loaded. Stop u-boot at the end stage as discussed earlier.

```
=====
Writing kernel image
=====
tftp 0x82000000 ulmage
sf probe
```



```
sf erase 0x100000 0x500000
sf write 0x82000000 0x100000 0x500000
```

```
=====
Writing ramdisk (file system) image
=====
```

```
tftp 0x82600000 ramdisk
sf probe
sf erase 0x600000 0x1400000
sf write 0x82600000 0x600000 0x1400000
```

```
=====
Writing device tree image
=====
```

```
tftp 0x88000000 imx6sl-tixu-ldo.dtb
sf probe
sf erase 0x2000000 0x20000
sf write 0x88000000 0x2000000 0x20000
```

##NOTE: In these commands, names and memory ranges may vary later based on image name and size.

1.1.5 Setting environment variables

After writing all images into flash, we have to set environment variables according to written images. Based on these environment variables only, u-boot will load all other images.

```
setenv bootargs 'console=ttyMXC0,115200 root=/dev/ram rw ramdisk_size=21000'
Setenv loadimages 'sf probe;sf read 0x82600000 0x600000 0x1400000;sf read 0x88000000
0x2000000 0x20000;sf read 0x82000000 0x100000 0x500000'
Setenv bootcmd 'run loadimages;bootm 0x82000000 0x82600000 0x88000000'
saveenv
```

```
=====
```

After setting all environment variables reboot the board by power button Give input for username as root.

```
root@imx6sltixu:~#.
```

5.3 SD Card boot mode

In this mode, board will boot from the images on SD card.

4.3.1 Format the SD card

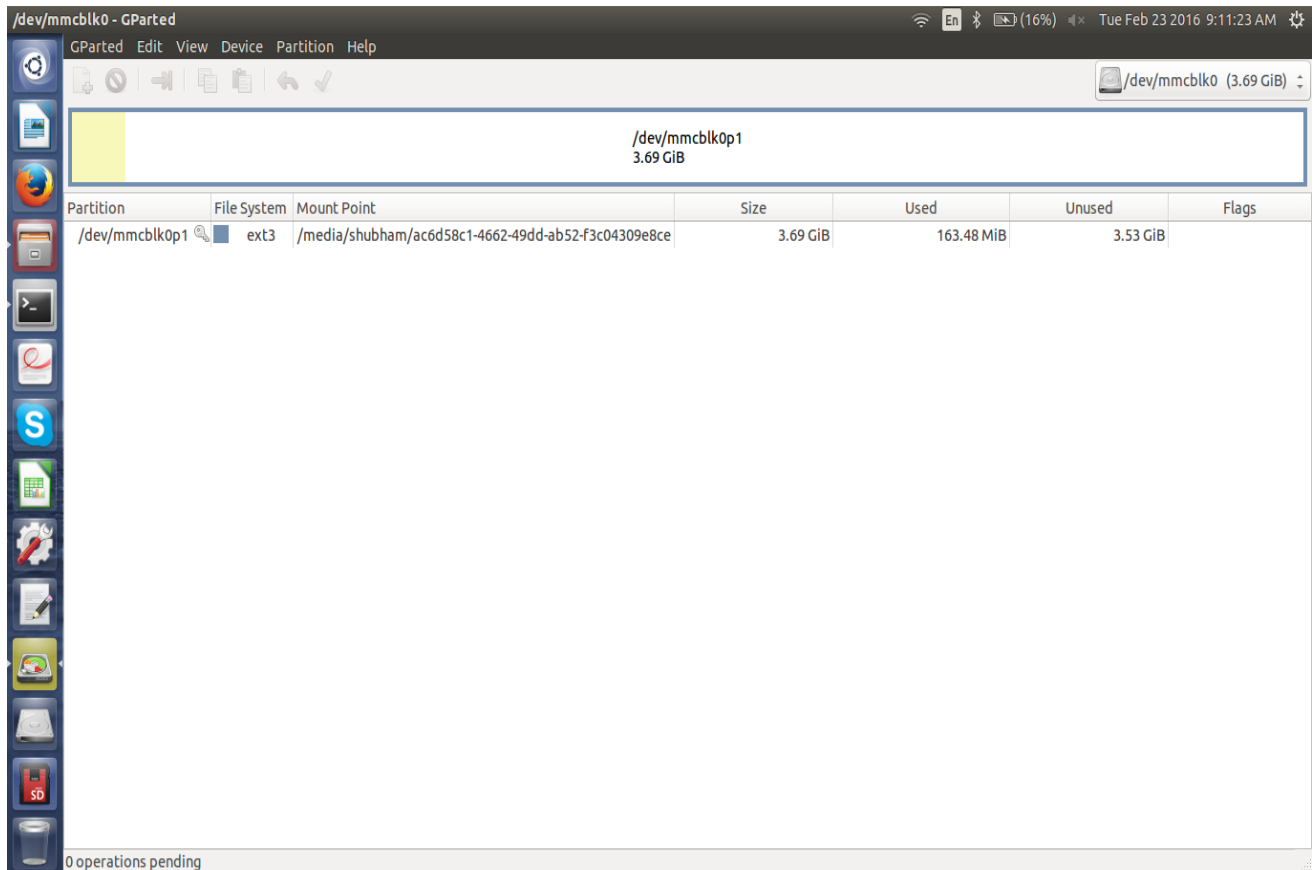
Format the SD card using gparted application.

To install gparted application, run the following command:

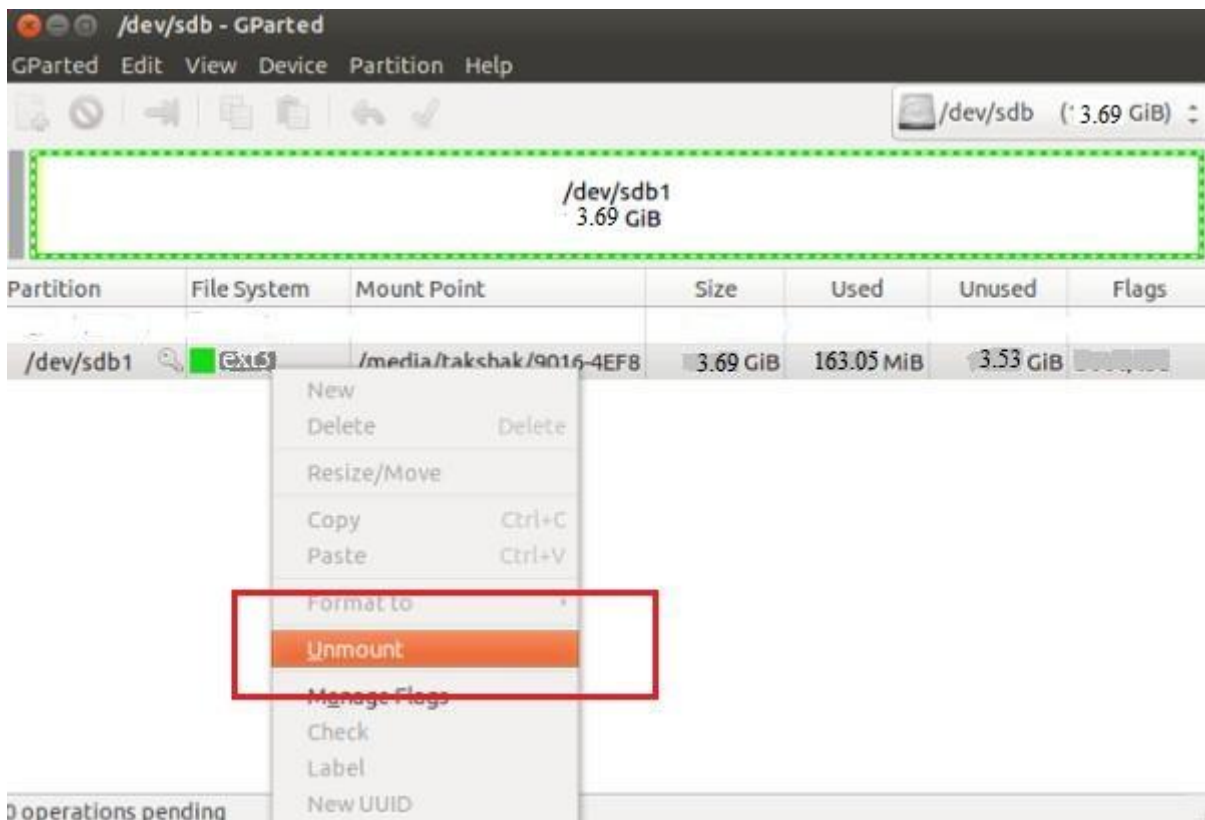
- `$ sudo apt-get install gparted`

After installing gparted, run the following command to format the SD card

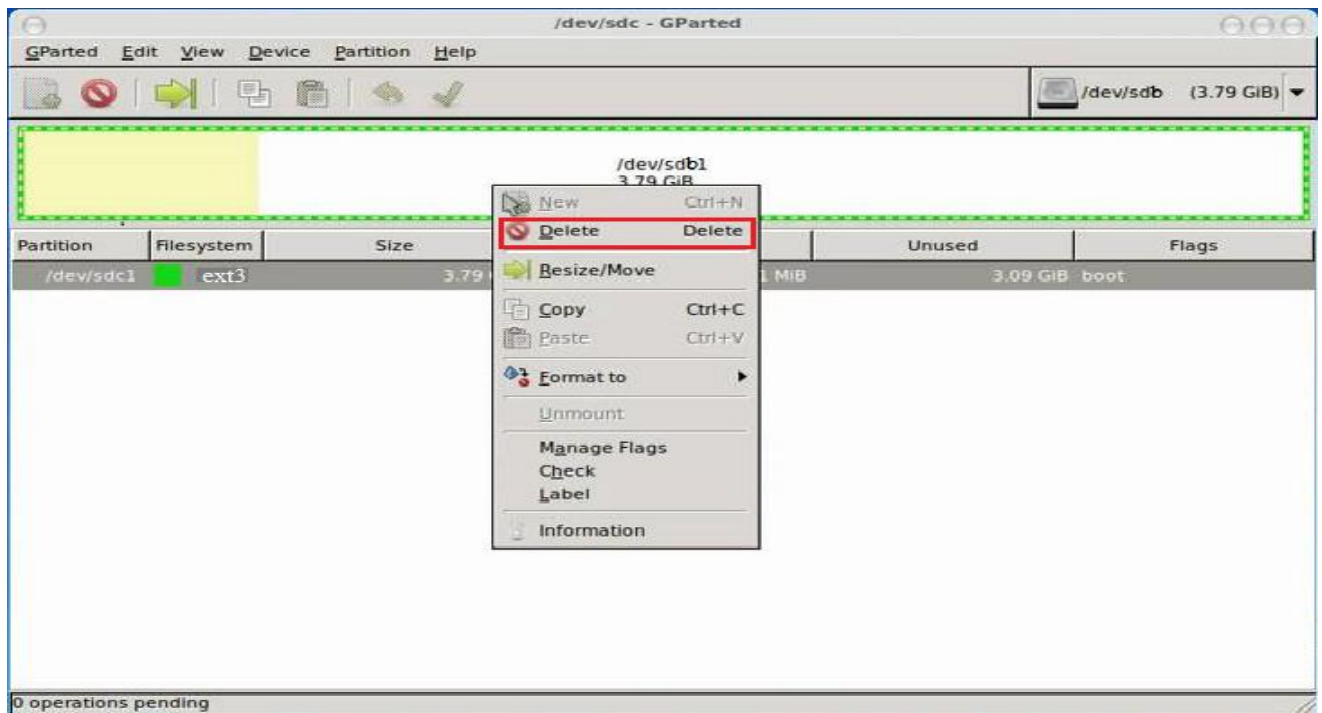
- `$ sudo gparted`



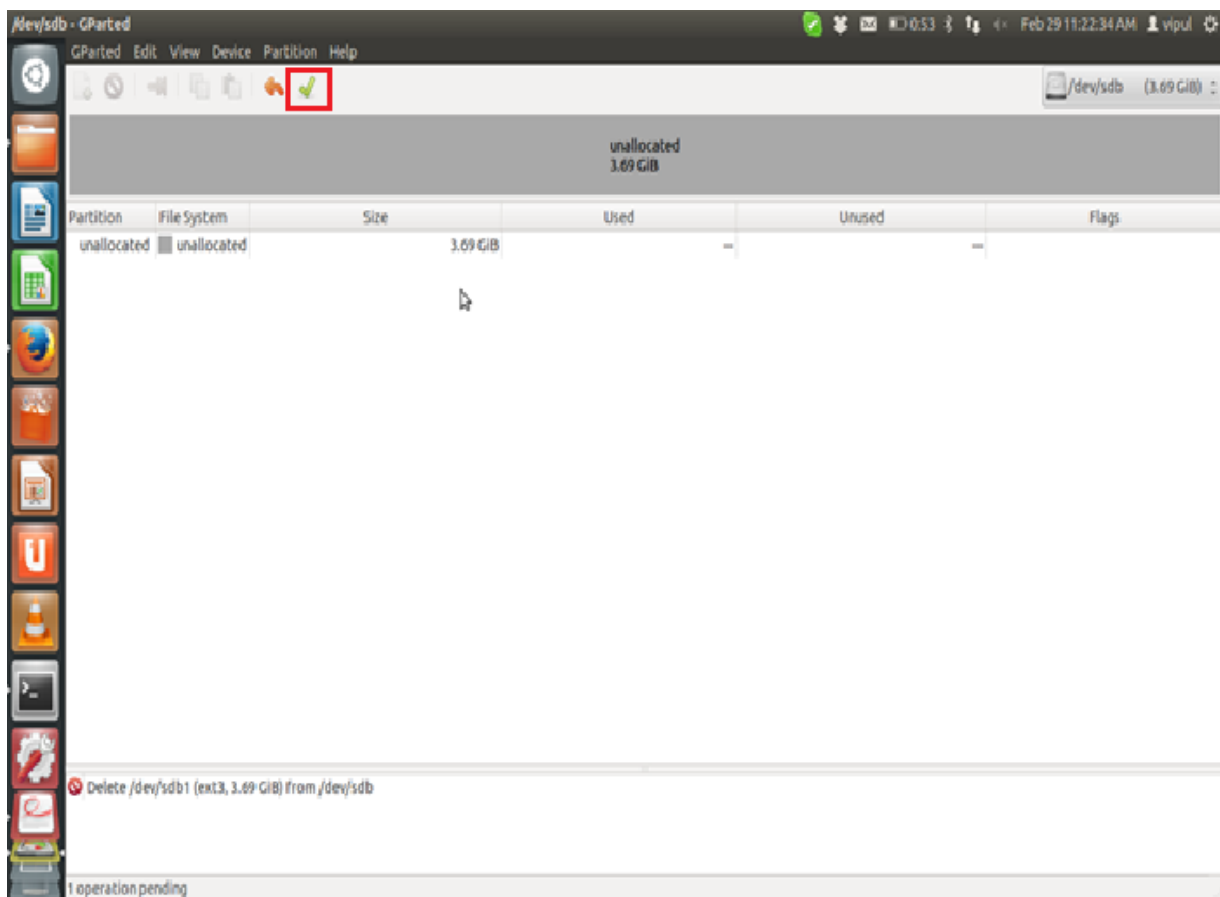
Now unmount the SD card as shown below

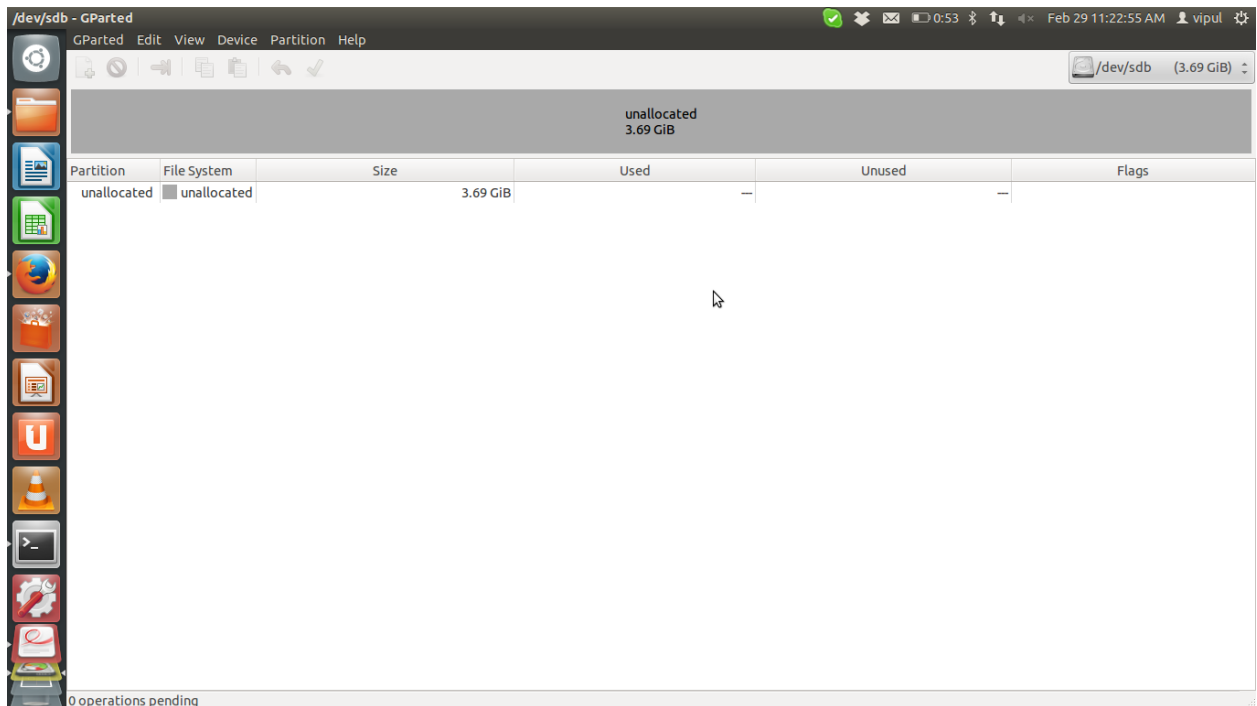


After unmounting the SD card, delete the partition as shown below



Now click on the button shown below





After formatting the SD card, remove the card and remount it again, after that go to the image directory for example

- `$ cd /home/<user>/tixu_pmic/src/build/tmp/deploy/images/imx6sltixu`
- `sudo dd if=core-image-minimal-imx6sltixu-xxxx.sdcard of=/dev/<sd card device> bs=1M && sync`

Sort the pins 1 and 2, 3 and 4 of the header H31.

Reboot the board to boot from SD card.

6. LEDs Notifications

LED	Interface Indication	COLOUR
D4	Wi-Fi (Client mode)	Green
D4	Wi-Fi (AP mode)	Green (blinking)
D5	ZigBee LED	Yellow
D6	BT LED	Orange
D4 and D5	Zigbee to Wlan Bridging	Yellow (Zigbee) and Green (Wi-Fi (blinking))
D4 and Ethernet LED	wlan to Ethernet Bridging	Green (Wi-Fi (blinking)) and Ethernet LED
DA2	RESET LED	Red

7. Wi-Fi Interface

7.1 Client mode

Following command will invoke the wlan0 interface

- `ifconfig wlan0 up`

```
wlan0 Link encap:Ethernet HWaddr 4c:bb:58:de:5a:fe
      inet addr:192.168.41.184 Bcast:192.168.43.255 Mask:255.255.252.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:82574 errors:0 dropped:0 overruns:0 frame:0
      RX bytes:11217279 (11.2 MB) TX bytes:794355 (794.3 KB)
```

Scan the AP's in order to connect one.

- `iw wlan0 scan | grep "SSID"`

Connect to any secured selected SSID.

- `wpa_passphrase <AP_NAME> <PASSWORD> > /etc/wpa_supplicant.conf`

To be able to connect to a remote AP, wpa_supplicant daemon must run to provide WPA key negotiation with a WPA Authenticator and EAP authentication with Authentication Server.

- `wpa_supplicant -Dnl80211 -iwlan0 -c /etc/wpa_supplicant.conf -B`

Request the dynamic IP.

- `udhcpc -i wlan0`

Connect to Open mode AP

Scan the AP's around in order to connect the desired one

- `$ iw wlan0 scan | grep SSID`

In order to connect to AP use the command

- `$ iw wlan0 connect <AP SSID>`

We can check the connection by invoking

- `iw wlan0 link`

Request the dynamic IP.

- `udhcpc -i wlan0`



Terminate the connection and Disable the Wi-Fi.

- `wifi_stop.sh`

6.1 AP mode

Run the following script to enable the WLAN in AP mode

- `tixu_AP_en.sh`



It will ask whether the user wants to create the AP in secured mode or open mode enter 1 to create the AP in secured mode and enter 2 to create AP in open mode.

```
wlcore: PHY firmware version: Rev 8.2.0.0.232
wlcore: firmware booted (Rev 8.9.0.0.48)
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
press 1-secure mode 2-open mode
```

It will create a Wi-Fi access point (AP) named TlxU_PMIC.

While searching for Wi-Fi networks, if the user has created the AP in secure mode then its default password is: 123456789.

If user wants to change the name of ssid and password then open hostapd.conf

```
vi /etc/hostapd.conf
ssid=<new SSID>
wpa_passphrase=<new paswd>
```

To close the Wi-Fi in AP mode, run the following command

```
wifi_stop.sh
```

6.2 WLAN to Ethernet Bridging

Run the following command to create the bridge

```
wlan-eth_bridge_create.sh
```

It will ask whether the user wants to create the AP in secured mode or open mode enter 1 to create the AP in secured mode and enter 2 to create AP in open mode.

```
wlcore: PHY firmware version: Rev 8.2.0.0.232
wlcore: firmware booted (Rev 8.9.0.0.48)
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
press 1-secure mode 2-open mode
```

It will create the bridge interface named **wlan0-bg**.

To check the bridge interface, run ifconfig

```
wlan0-bg  Link encap:Ethernet HWaddr 00:50:C2:BC:C0:F1
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:328 (328.0 B) TX bytes:1296 (1.2 KiB)
```

To Remove the Bridge interface

```
ifconfig wlan0-bg down
brctl delbr wlan0-bg
wifi_stop.sh
```

6.3 Wi-Fi Throughput

TCP throughput: 94 Mb/s.

```
iperf -s -i2
```

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

```
-----
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 57301
[ ID] Interval      Transfer    Bandwidth
[  4]  0.0- 2.0 sec  21.7 MBytes 91.0 Mbits/sec
[  4]  2.0- 4.0 sec  22.4 MBytes 94.1 Mbits/sec
[  4]  4.0- 6.0 sec  22.4 MBytes 93.9 Mbits/sec
[  4]  6.0- 8.0 sec  22.4 MBytes 94.1 Mbits/sec
[  4]  8.0-10.0 sec  22.4 MBytes 94.0 Mbits/sec
[  4] 10.0-12.0 sec  22.4 MBytes 94.1 Mbits/sec
[  4] 12.0-14.0 sec  22.4 MBytes 94.1 Mbits/sec
[  4] 14.0-16.0 sec  22.4 MBytes 93.8 Mbits/sec
```

```
root@imx6sltixu:~# iperf -c 192.168.1.100 -i2 -t30
```

Client connecting to 192.168.1.100, TCP port 5001

TCP window size: 43.8 KByte (default)

```
-----
[  3] local 192.168.1.10 port 57301 connected with 192.168.1.100 port 5001
[ ID] Interval      Transfer    Bandwidth
[  3]  0.0- 2.0 sec  22.1 MBytes 92.8 Mbits/sec
[  3]  2.0- 4.0 sec  22.9 MBytes 95.9 Mbits/sec
[  3]  4.0- 6.0 sec  22.6 MBytes 94.9 Mbits/sec
[  3]  6.0- 8.0 sec  22.4 MBytes 93.8 Mbits/sec
[  3]  8.0-10.0 sec  22.4 MBytes 93.8 Mbits/sec
[  3] 10.0-12.0 sec  23.0 MBytes 96.5 Mbits/sec
[  3] 12.0-14.0 sec  23.0 MBytes 96.5 Mbits/sec
[  3] 14.0-16.0 sec  22.2 MBytes 93.3 Mbits/sec
[  3] 16.0-18.0 sec  22.6 MBytes 94.9 Mbits/sec
[  3] 18.0-20.0 sec  22.2 MBytes 93.3 Mbits/sec
```


8. Bluetooth Interface

Run following script to enable the BT interfaces

```
bt_start.sh
```

To check the BT interface run following command

```
hciconfig hci0
```

```
hci0:    Type: BR/EDR   Bus: UART
        BD Address: 98:7B:F3:CF:CC:6F  ACL MTU: 1021:6  SCO MTU: 180:4
        UP RUNNING PSCAN ISCAN
        RX bytes:2483598 acl:2615 sco:0 events:407 errors:0
        TX bytes:7379 acl:236 sco:0 commands:151 errors:0
```

Scan the devices in vicinity:

- ```
hcitool scan
```

Pair with selected device:

- ```
bt-device -c <bt_addr>
```

To see the paired devices run the following command

- ```
bt-device -l
```

For non-android non-windows mobiles:

Run the following command to transfer files to remote BT device

- ```
obexftp -b <remote_bt_addr> -p <file_name>
```

For removing the paired device

- ```
bt-device -r <BD_addr>\
```

To see the channel number of the remote BT to send file to that remote device, run the following command.

- ```
sdptool browse <bt_add>
```

```
Service Name: OBEX Object Push
```

```
Service RecHandle: 0x1000b
```

```
Service Class ID List:
```

```
"OBEX Object Push" (0x1105)
```

```
Protocol Descriptor List:
```

```
"L2CAP" (0x0100)
```

```
"RFCOMM" (0x0003)
```

```
Channel: 25
```

```
"OBEX" (0x0008)
```

```
Profile Descriptor List:
```

```
"OBEX Object Push" (0x1105)
```

```
Version: 0x0100
```

Transfer files to any android/windows mobile

- ```
obexftp --uuid none -b <BT_addr> -B <channel No.> -p <file path>
```

```
example: obexftp --uuid none -b FC:64:BA:06:B1:F4 -B 12 -p File.txt
```

<BT\_addr>=Bluetooth mac address of remote device

<file path>=File with exact path to be send

<channel No.>=OBEX Object Push service channel number

To close the BT interface

- `bt_stop.sh`

## 9. ZigBee Interface

To check ZigBee interface open two serial consoles of the board (one uart console or 2 dedicated telnet accesses). (Board default IP is 192.168.1.200).

On the first terminal, change the directory to `/home/root/zigbee/servers`.

```
cd zigbee/servers
```

Run the script using following command

```
./zigbeeHAgw bbb > /dev/null 2&>1
```

On the second terminal, change the directory to `/home/root/zigbee/servers`.

```
cd zigbee/servers
```

Run the Zigbee testing application using the command

```
./start_zigbeeApp
```

Now a Demo app will be displayed which can be used to view the ZigBee devices and control them.

To see more HELP type `'?'`.

Enter `"p"` to start searching for end device

After the device is connected press UP/DOWN arrow to select that device and then enter `"n"` to turn ON the end device and `"f"` to turn OFF the end device.

To close the zigbee interfaces

Enter `"q"` on the terminal on which zigbee application is running and enter `ctrl+c` on the other terminal.

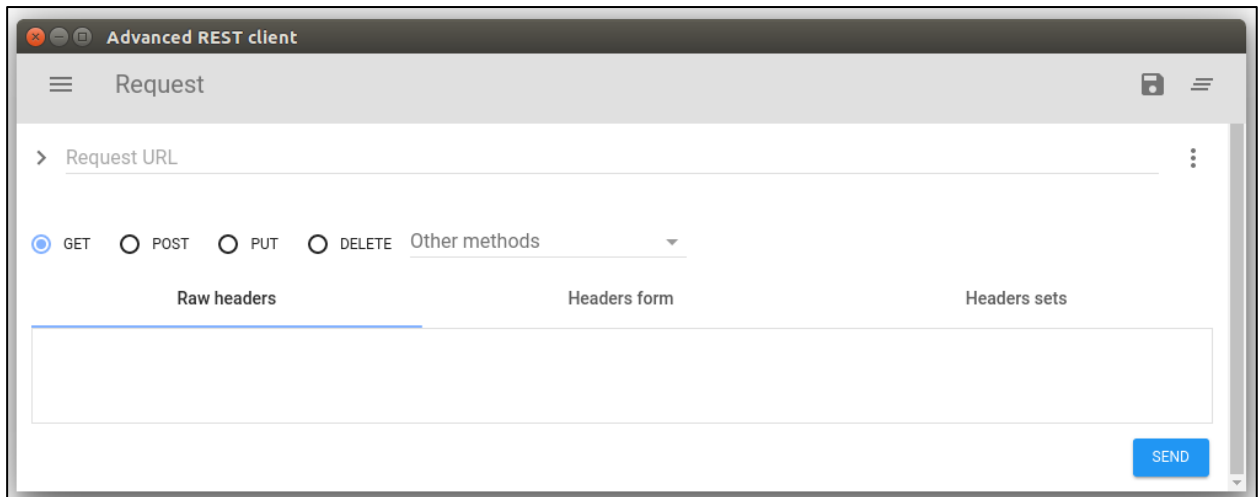
## 9.1 ZigBee to WLAN Bridging

Run the following script file

- `zigbee_wlan_bridge.sh`

This script will produce output similar to image below

Open Chrome browser and add Advanced Rest Client extension to it.



```
alkesh@alkesh: ~
*** TI GATEWAY DEMO APP v1.00 *****
=NETWORK=====
EXTPANID:00:4B:12:00:AB:A9:00:06 PANID:0x42DF CHANNEL:11 STATE: ACTIVE,CLOSED
=[DEVICES]=====
IEEE_ADDR ADDR EP DETAILS | EP DETAILS | ...
06:0D:A9:AB:00:12:4B:00 0000 02 0007(HA) | 03 0100(HA) | 04 0101(HA) | 05 0102(HA) | 06 0102(HA)

=GROUPS=====
GROUP_0 group_1 group_2 group_3 group_4 group_5 group_6 group_7 group_8 group_9
=SCENES=====
SCENE_0 scene_1 scene_2 scene_3 scene_4 scene_5 scene_6 scene_7 scene_8 scene_9
=ACTIONS=====
(UNICAST)=====
[OFF\ON] [LVL:250] [HUE:250] [SAT:254] [PERMIT_JOIN:060] [REMOVE] [BIND\UNBIND]
[GROUP_ADD\REMOVE] [SCENE_STORE\REMOVE] [SCENE_RECALL] [QUIT]
=LOG=====

head: invalid option -- '1'
BusyBox v1.23.2 (2016-04-25 18:04:33 IST) multi-call binary.

Usage: head [OPTIONS] [FILE]...

head: invalid option -- '1'
BusyBox v1.23.2 (2016-04-25 18:04:33 IST) multi-call binary.

Usage: head [OPTIONS] [FILE]...

head: invalid option -- '1'
BusyBox v1.23.2 (2016-04-25 18:04:33 IST) multi-call binary.

Usage: head [OPTIONS] [FILE]...

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

To add new node select GET and write in Request URL:

```
http://<IP>/zbha/<user>/node/new
```

<user> is by default user1

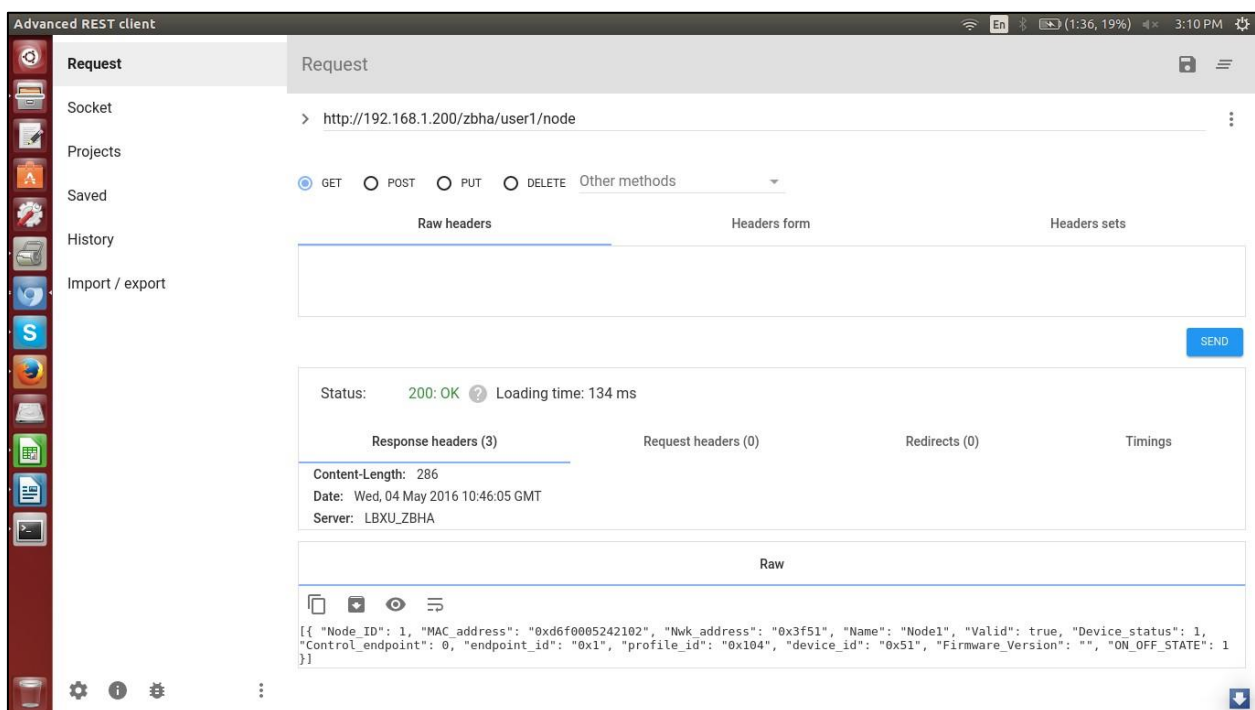
This make zigbee server to open state so that it can connect to end devices

If an end device is connected to the zigbee server it will look like this.

[illegible]

To check status of device select GET and write in Request URL :

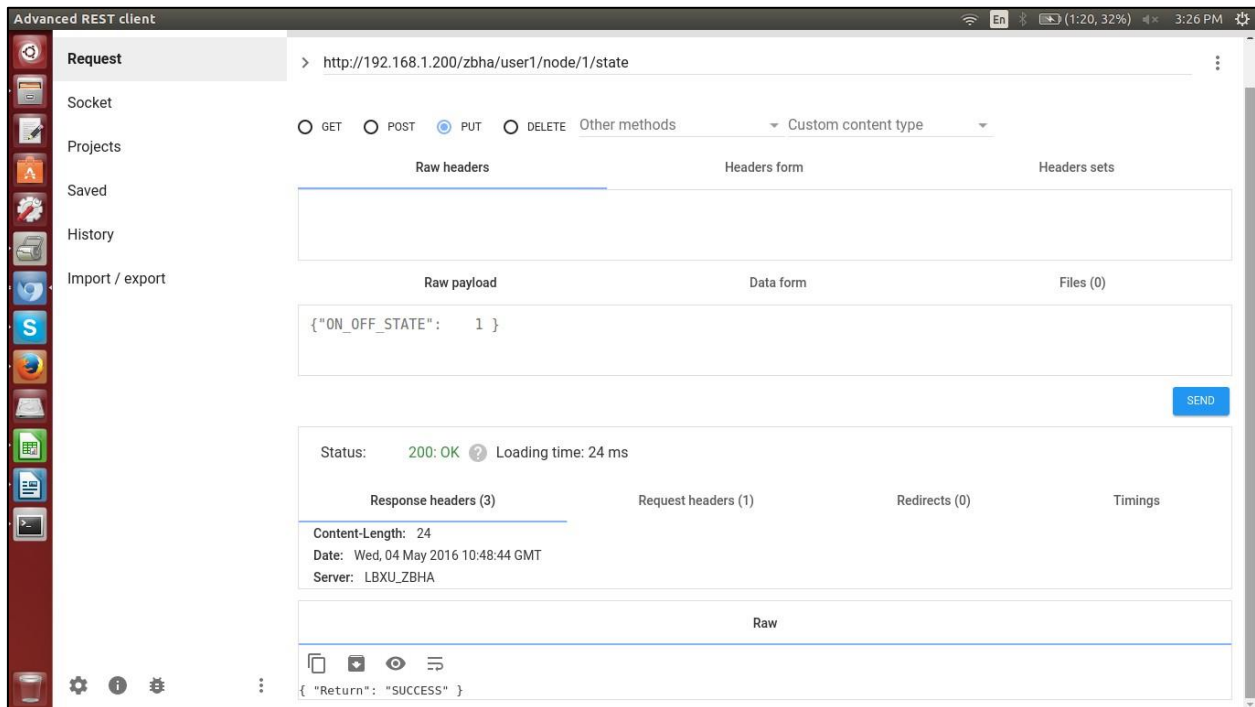
- `http://<IP>/zbha/<user>/node`



To change ON/OFF state of device select PUT and write in Request URL:

http://<IP>/zbha/<user>/node/1/state

Write in Raw payload: - {"ON\_OFF\_STATE": 1}



- Device will be ON if { "ON\_OFF\_STATE" : 1 }
- Device will be OFF if { "ON\_OFF\_STATE" : 0 }
- We can perform many more functions through API.
- Refer excel file TIXU\_PMIC\_HIGHEND\_GATEWAY\_HTTP\_API.xlsx for more functions

To close the zigbee to wlan interface run following command

- `zigbee_stop.sh`

## 10. Ethernet Interface

### 10.1 Interface test

Run following command to perform throughput test for Ethernet.

Bring the Ethernet interface up after connecting the LAN cable

- `$ ifconfig eth0 up`

### 10.2 Throughput

**TCP throughput: 94.2 Mb/s**

Run following command on SERVER side (on our board)

- `root@imx6sltixu:~# iperf -s -i2`

This will produce the output similar to the following

Server listening on TCP port 5001

TCP window size: 320 KByte (WARNING: requested 2.00 MByte)

```

[4] local 192.168.1.200 port 5001 connected with 192.168.1.100 port 50139
[ID] Interval Transfer Bandwidth
[4] 0.0- 1.0 sec 11.3 MBytes 94.6 Mbits/sec
[4] 1.0- 2.0 sec 11.2 MBytes 94.2 Mbits/sec
[4] 2.0- 3.0 sec 11.2 MBytes 94.2 Mbits/sec
[4] 0.0- 3.3 sec 36.8 MBytes 94.3 Mbits/sec
[5] local 192.168.1.200 port 5001 connected with 192.168.1.100 port 50141
[5] 0.0- 1.0 sec 11.3 MBytes 94.5 Mbits/sec
[5] 1.0- 2.0 sec 11.2 MBytes 94.2 Mbits/sec
```

Run following command on the CLIENT side (on our PC)

- `iperf -c 192.168.1.200 -i2 -t14`

This will produce the output similar to following

```

Client connecting to 192.168.1.200, TCP port 5001
TCP window size: 85.0 KByte (default)

[3] local 192.168.1.100 port 50141 connected with 192.168.1.200 port 5001
[ID] Interval Transfer Bandwidth
[3] 0.0- 2.0 sec 22.6 MBytes 94.9 Mbits/sec
[3] 2.0- 4.0 sec 22.4 MBytes 93.8 Mbits/sec
[3] 4.0- 6.0 sec 22.5 MBytes 94.4 Mbits/sec
[3] 6.0- 8.0 sec 22.5 MBytes 94.4 Mbits/sec
[3] 8.0-10.0 sec 22.5 MBytes 94.4 Mbits/sec
[3] 10.0-12.0 sec 22.4 MBytes 93.8 Mbits/sec
[3] 12.0-14.0 sec 22.4 MBytes 93.8 Mbits/sec
```

## 11. SD Card Interface

Connect u-SD card to card slot.

Read/Write performance:

- `hdparm -t /dev/mmcblk0`

The above command produces the following output

```
/dev/mmcblk0:
Timing buffered disk reads: 50 MB in 3.03 seconds = 16.51 MB/sec
```

## 12. TPS65910 PMIC Interface

### 12.1 Low-level Power Management (PM)

Information found here describes the low-level Power Management (PM) driver which controls the low-power modes.

The i.MX6SL supports four low power modes: RUN, WAIT, STOP, and DORMANT.

Table below lists the detailed clock information for the different low power modes.

| Mode    | Core      | Modules                 | PLL | CKIH/FPM | CKIL |
|---------|-----------|-------------------------|-----|----------|------|
| RUN     | Active    | Active, Idle or Disable | On  | On       | On   |
| WAIT    | Disable   | Active, Idle or Disable | On  | On       | On   |
| STOP    | Disable   | Disable                 | Off | Off      | On   |
| DORMANT | Power off | Disable                 | Off | Off      | On   |

### 12.2 Software Operation

The i.MX 6 PM driver maps the low-power modes to the kernel power management states as listed below:

- **Standby-** maps to STOP mode which offers significant power saving, as all blocks in the system are put into a low-power state, except for ARM core, which is still powered on, and memory is placed in self-refresh mode to retain its contents.
- **Mem (suspend to RAM)** - maps to DORMANT mode which offers most significant power saving as all blocks in the system are put into a low-power state, except for memory, which is placed in self-refresh mode to retain its contents • System idle which maps to WAIT mode
- **Freeze (low -power idle)** - This state is a generic, pure software, light-weight, system sleep state. It allows more energy to be saved relative to runtime idle by freezing user space and putting all I/O devices into low-power states (possibly lower-power than available at run time), such that the processors can spend more time in their idle states.

The i.MX 6 PM driver performs the following steps to enter and exit low power mode:

1. Allow the Cortex-A9 platform to issue a deep sleep mode request.
2. If STOP or DORMANT mode.
  - Program CCM CLPCR register to set low power control register.
  - If DORMANT mode, request switching off CPU power when pdn\_req is asserted.
  - Request switching off embedded memory peripheral power when pdn\_req is asserted.
  - Program GPC mask register to unmask wakeup interrupts.
3. Call CPU\_do\_idle to execute WFI pending instructions for wait mode.
4. Execute imx6\_suspend in IRAM.



5. If in DORMANT mode, save ARM context, change the drive strength of MMDC PADs as "low" to minimize the power leakage in DDR PADs. Execute WFI pending instructions for stop mode.
6. Generate a wakeup interrupt and exit low power mode. If DORMANT mode, restore
7. ARM core and DDR drive strength.

In DORMANT mode, the i.MX 6 can assert the VSTBY signal to the PMIC and request a voltage change.

To enter different system level low power modes:

```
echo mem > /sys/power/state
```

```
echo standby > /sys/power/state
```

### 12.3 Wakeup from low-power modes

To wake up system from low power modes, enable the wakeup source first like debug UART or RTC, which can be used as a wakeup source.

Below is the example of UART wakeup:

```
echo enabled > /sys/bus/platform/drivers/imx-uart/`xxxxxxx`.serial/tty/ttymxc'y'/power/wakeup
```

Here 'xxxxxxx' is the physical base address of your debugging UART. For example, for UART1, it is 2020000. 'y' is our debugging UART index.

### 12.4 Dynamic Voltage Frequency Scaling Operation

CPU frequency scaling enables the operating system to scale the CPU frequency up or down in order to save power. CPU frequencies can be scaled automatically depending on the system load, in response to ACPI events, or manually by user space programs.

The CPU frequency scaling device driver allows the clock speed of the CPU to be changed on the fly. Once the CPU frequency is changed, the voltage VDDCORE, VDDSOC and VDDPU are changed to the voltage value defined in device tree scripts (DTS). This method can reduce power consumption (thus saving battery power), because the CPU uses less power as the clock speed is reduced.

Supported frequencies to be scaled:

- 396 MHz
- 792 MHz
- 996 MHz

## DVFS Driver Architectural Overview

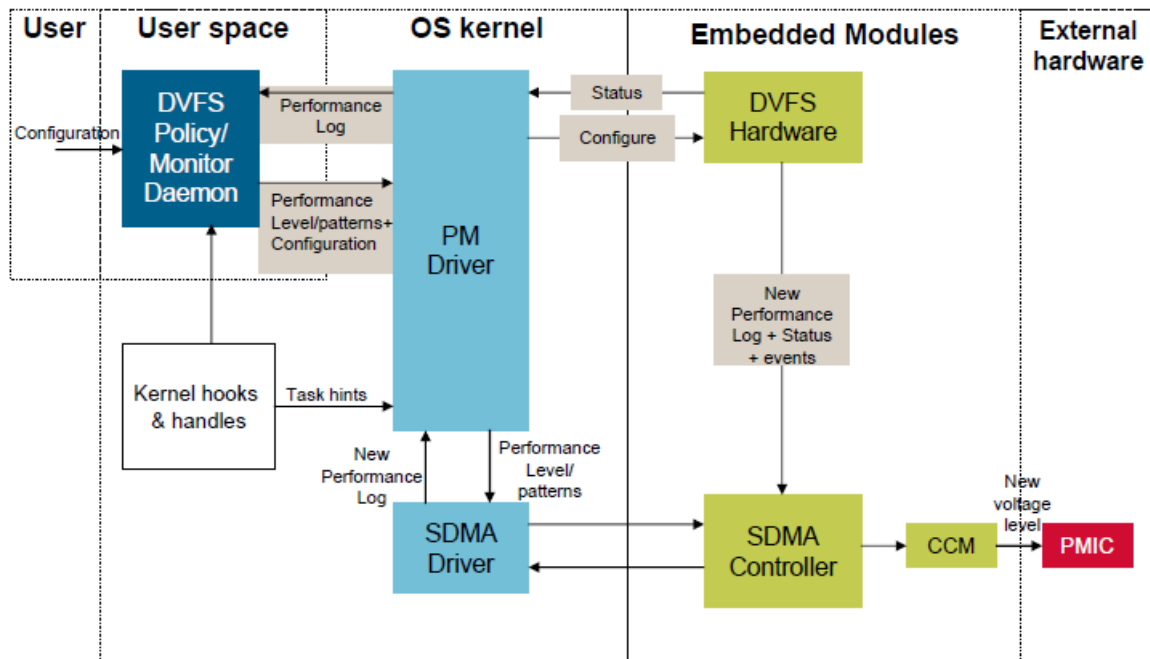


Figure 3 DVFS architecture

TPS65910 offers very wide voltage range DVS capability and support decay mode for all switched mode power rails. PMIC having two step-down converters with DVS (Dynamic voltage scaling) features; they are used to provide power for processor cores and are controllable by a dedicated class-3 smart Reflex interface for optimum Power savings.

A supply voltage value corresponding to a targeted frequency of the digital core supplied is programmed in VDD1\_OP\_REG & VDD2\_OP\_REG registers. These registers can be access through dedicated serial control interface (SR-I2C). This control interface is complaint with HS-I2C specification.

To optimize power efficiency on gateway board, the voltage domain of iMX6SL uses the DVFS feature provided by PMIC. If processor needs to run at higher frequency, corresponding supplied power will be also increased using DVFS functionality.

Below Table describe used converter for specific power rail of processor.

| PMIC | Features | iMx6 Power Rail | Voltage Range |
|------|----------|-----------------|---------------|
| SW1  | DVFS     | VDD_ARM_IN      | 1.375 – 1.5V  |
| SW2  | DVFS     | VDD_SoC_IN      | 1.275 – 1.5V  |

Available Governors in the Linux Kernel:

| Governor     | Detail                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Performance  | The CPUfreq governor "performance" sets the CPU statically to the highest frequency within the borders of scaling_min_freq and scaling_max_freq.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Powersave    | The CPUfreq governor "powersave" sets the CPU statically to the lowest frequency within the borders of scaling_min_freq and scaling_max_freq.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Userspace    | The CPUfreq governor "user space" allows the user, or any user space program running with UID "root", to set the CPU to a specific frequency by making a sysfs file "scaling_setspeed" available in the CPU-device directory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Ondemand     | The CPUfreq governor "ondemand" sets the CPU depending on the current usage. To do this the CPU must have the capability to switch the frequency very quickly.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Conservative | The CPUfreq governor "conservative", much like the "ondemand" governor, sets the CPU depending on the current usage. It differs in behavior in that it gracefully increases and decreases the CPU speed rather than jumping to max speed the moment there is any load on the CPU. This behavior more suitable in a battery powered environment.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Interactive  | <p>The CPUfreq governor "interactive" is designed for latency-sensitive, interactive workloads. This governor sets the CPU speed depending on usage, similar to "ondemand" and "conservative" governors. However, the governor is more aggressive about scaling the CPU speed up in response to CPU-intensive activity. Sampling the CPU load every X ms can lead to under-powering the CPU for X ms, leading to dropped frames, stuttering UI, etc. Instead of sampling the CPU at a specified rate, the interactive governor will check whether to scale the CPU frequency up soon after coming out of idle. When the CPU comes out of idle, a timer is configured to fire within 1-2 ticks. If the CPU is very busy between exiting idle and when the timer fires then we assume the CPU is underpowered and ramp to MAX speed.</p> <p>If the CPU was not sufficiently busy to immediately ramp to MAX speed, then governor evaluates the CPU load since the last speed adjustment, choosing the highest value between that longer-term load or the short-term load since idle exit to determine the CPU speed to ramp to.</p> |

### 1.1.6 Software Operations

To view what values the CPU frequency can be changed to in KHz (The values in the first column are the frequency values) use this command:

```
cat /sys/devices/system/CPU/CPU0/CPUfreq/stats/time_in_state
```

To change the CPU frequency to a value that is given by using the command above (for example, to 792 MHz) use this command,

```
echo 792000 > /sys/devices/system/CPU/CPU0/CPUfreq/scaling_setspeed
```



The frequency 792000 is in KHz, which is 792 MHz. The maximum frequency can be checked using this command:

```
cat /sys/devices/system/CPU/CPU0/CPUfreq/scaling_max_freq
```

Use the following command to view the current CPU frequency in KHz:

```
cat /sys/devices/system/CPU/CPU0/CPUfreq/CPUinfo_cur_freq
```

Use the following command to view available governors:

```
cat /sys/devices/system/CPU/CPU0/CPUfreq/scaling_available_governors
```

Use the following command to change to interactive CPU frequency governor:

```
echo powersave > /sys/devices/system/CPU/CPU0/CPUfreq/scaling_governor
```

## 13. Board Button Functionalities

There are two buttons present on the TIXU board.

| Button | Board Name | Functionality                    |
|--------|------------|----------------------------------|
| SW2    | SYS RST    | Resets the board on single press |
| SW3    | CPU RST    | Board Power ON button            |

## 14. Dhrystone Benchmark

Dhrystone benchmark test will be performed to measure the integer performance of processors and compilers. Dhrystone is a very small integer benchmark that must be run multiple times in order to obtain reproducible numbers.

### 14.1 Dhrystone Performance Calculation

Dhrystone performance is calculated using this formula:

**Dhrystone per second = number of runs / execution time**

For the result to valid, the Dhrystone code must be executed for at least two seconds, although longer is generally is better, and ARM recommended at least 20 seconds. The official Dhrystone source was released on the Usenet forum in 1988. There is no central repository containing it. You can access it from the following locations:

<http://fossies.org/linux/privat/old/dhrystone-2.1.tar.gz/>

One common representation of the Dhrystone benchmark is DMIPS. DMIPS (Dhrystone MIPS). It is obtained when the Dhrystone score is divided by **1757** (the number of Dhrystones per second obtained on the VAX 11/780, nominally a 1 MIPS machine).

To run Dhrystone Benchmark run

```
gcc_dry2reg
```

The number of runs through the benchmark should be 1000000 for frequency >= 1GHz

Dhrystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Please give the number of runs through the benchmark: 600000

Execution starts, 600000 runs through Dhrystone

Execution ends

Final values of the variables used in the benchmark:

Int\_Glob: 5

should be: 5

Bool\_Glob: 1

should be: 1

Ch\_1\_Glob: A

should be: A

Ch\_2\_Glob: B

should be: B

Arr\_1\_Glob[8]: 7

should be: 7

Arr\_2\_Glob[8][7]: 600010

should be: Number\_Of\_Runs + 10

Ptr\_Glob->

Ptr\_Comp: 14815240

should be: (implementation-dependent)

Discr: 0

should be: 0

Enum\_Comp: 2

should be: 2

Int\_Comp: 17

should be: 17

Str\_Comp: DHRYSTONE PROGRAM, SOME STRING

should be: DHRYSTONE PROGRAM, SOME STRING

Next\_Ptr\_Glob->

Ptr\_Comp: 14815240

should be: (implementation-dependent), same as above

Discr: 0

should be: 0

Enum\_Comp: 1

should be: 1



```

Int_Comp: 18
 should be: 18
Str_Comp: DHRYSTONE PROGRAM, SOME STRING
 should be: DHRYSTONE PROGRAM, SOME STRING
Int_1_Loc: 5
 should be: 5
Int_2_Loc: 13
 should be: 13
Int_3_Loc: 7
 should be: 7
Enum_Loc: 1
 should be: 1
Str_1_Loc: DHRYSTONE PROGRAM, 1'ST STRING
 should be: DHRYSTONE PROGRAM, 1'ST STRING
Str_2_Loc: DHRYSTONE PROGRAM, 2'ND STRING
 should be: DHRYSTONE PROGRAM, 2'ND STRING

```

```

Microseconds for one run through Dhrystone: 23.2
Dhrystones per Second: 43113.8

```

So, DMIPS = 43113.8/1754 = 24.538

## 15. RTC

The RTC, which is driven by the 32-kHz clock, provides the alarm and timekeeping functions. The RTC is kept supplied when the device is in the OFF or the BACKUP state.

The main functions of the RTC block are:

- Time information (seconds/minutes/hours) directly in binary-coded decimal (BCD) format
- Calendar information (Day/Month/Year/Day of the week) directly in BCD code up to year 2099
- Programmable interrupts generation: The RTC can generate two interrupts: a timer interrupt

RTC\_PERIOD\_IT periodically (1s/1m/1h/1d period) and an alarm interrupt RTC\_ALARM\_IT at a precise time of the day (alarm function). These interrupts are enabled using IT\_ALARM and IT\_TIMER control bits. Periodically interrupts can be masked during the SLEEP period to avoid host interruption and are automatically unmasked after SLEEP wakeup (using the IT\_SLEEP\_MASK\_EN control bit).

- Oscillator frequency calibration and time correction

To use TPS65910 RTC as a wake-up source, use the following procedure:

**Step 1:** Set the date to the current date and time

```
date -s "yyyy-mm-dd hh:mm:sec"
```

**Step 2:** Set the hwclock to the current date and time

```
hwclock -systohc
```

**Step 3:** Now set the alarm time

```
echo +x > /sys/class/rtc/rtc0/wakealarm
```

It will wake up the board after "x" seconds

**Step 4:** Now put the board in sleeping mode

```
echo standby > /sys/power/state
```

To use SNVS RTC as a wakeup source, enable it from the defconfig, it will register snvs as rtc0. Use the following commands:

```
echo +x > /sys/class/rtc/rtc0/wakealarm
```

```
echo standby > /sys/power/state
```

RTC will wake up system after 'x' seconds.

Similarly, we can use RTC to wake up system from different low power modes.