RPU Debug Infrastructure  
Triage

Copyright © Imagination Technologies Limited. All Rights Reserved.

‑‑This document is strictly confidential. Neither the whole nor any part of the information contained in, nor the product described in, this document may be adapted or reproduced in any material form except with the written permission of Imagination Technologies Limited. Imagination Technologies, the Imagination logo, PowerVR, MIPS, Meta, Ensigma and Codescape are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners. This document can only be distributed subject to the terms of a Non‑Disclosure Agreement or Licence with Imagination Technologies Limited.

Filename : RPU Debug Infrastructure.Triage.docx  
Version : 1.0.24 Not Issued - Live Document  
Issue Date : 05 Oct 2017  
Author : Imagination Technologies Limited

Contents

[Abbreviations 3](#_Toc494986693)

[1. Introduction 4](#_Toc494986694)

[2. Configuration 4](#_Toc494986695)

[3. Statistics 4](#_Toc494986696)

[4. Gathering stats 5](#_Toc494986697)

[5. Run time Debug Support 6](#_Toc494986698)

[5.1. Module Loading Time 7](#_Toc494986699)

[5.2. Run Time 7](#_Toc494986700)

[5.3. Collecting run time logs 8](#_Toc494986701)

[6. Post Processing 9](#_Toc494986702)

List of Figures

**No table of figures entries found.**

IF{{Internal}}

Document History

| Issue | Date | Changes/Comments |
| --- | --- | --- |
| 1.0.16 | 05 Oct 2017 | External Issue. |
| 1.0.19 | 05 Oct 2017 | External Issue |
| 1.0.24 | 05 Oct 2017 | External Issue |

END{{Internal}}

# Abbreviations

RPU - Radio Processing Unit

MAC - Medium Access Control

PHY - Physical Layer

HAL - Hardware Abstraction Layer

Procfs - Linux proc Pseudo Filesystem

IMG - Imagination

# Introduction

This document describes the debug methodology to be followed in case of any RPU WLAN related issues.

# Configuration

For any debug it's important to know the context of the issue and configuration in use, for RPU WLAN issues typically the below information is needed

1. Steps to Reproduce
2. AP Name and Model
3. Channel Information
   1. Channel Number: "iw wlan0" should give this info
   2. Channel BW: 20/40/80: "iw wlan0" should give this info
   3. Traffic Density: Clean/Moderate/Heavy interference (WLAN Sniffer can give this info)
4. Connection Type: AP's configuration should give this info, from DUT side "iw wlan0" should give this info
   1. Mode: 11b/a/g/n/ac
   2. Security: open/wpa-psk/wpa2-psk/wpa2-enteprise etc

# Statistics

The procfs of the host driver provides access to the RPU driver and firmware statistics. These comprise of HAL, MAC and PHY related counts.

1. Command to view PHY stats: cat /proc/rpu/phy\_stats
2. Command to view MAC stats: cat /proc/rpu/mac\_stats
3. Command to view HAL stats: cat /proc/rpu/hal\_stats
4. Command to retrieve the latest stats: echo get\_stats=1 > /proc/rpu/params

The last (#4) command should be executed each time view the latest stats.

# Gathering stats

Once the system is up, the stats can be gathered by periodically polling the /proc interface using a shell script running on the DUT.

An example script “get\_rpu\_stats.sh” is included in the package.

**Usage:** no arguments.

**# sh get\_rpu\_stats.sh &**

**Output:**

Creates a file called “rpu\_stats.txt”

Note: Please take back-up of “rpu\_stats.txt” before running the script again, the script removes any files with same name.

It polls the stats every 3 seconds for 10 times. The test case can be a data session or any other scenario in which the problem is seen. Once the stats have been gathered, the file “rpu\_stat.txt” should be shared with IMG for analysis of the issue.

If the test case is repeatable, then please run the above script once before starting the test case and once after. The number of trails can be edited in the script.

# Run time Debug Support

To debug host driver issues, we have a feature “run time debug” using which we can enable prints related to each module on the fly. This is helpful to collect logs directly from the field without the need for any re-compilation of the driver.

Currently we support the below modules for this feature, please find below the bitmap related to each module. The number beside each module represents the position of its bit, RPU\_DEBUG\_<MODULE\_NAME> = BIT (Bit position of the module)

Eg: RPU\_DEBUG\_CORE = BIT (4): This means that BIT4 is reserved for CORE module.

* RPU\_DEBUG\_SCAN = BIT(1)
  + For Debugging Scanning related issues.
* RPU\_DEBUG\_ROC = BIT(2)
  + For Debugging Remain on Channel (ROC) related issues. Useful debugging p2p\_find issues.
* RPU\_DEBUG\_TX = BIT(3)
  + For Debugging TX Data Path related issues.
  + Note: These debug’s can be huge.
* RPU\_DEBUG\_MAIN = BIT(4)
  + For Debugging Core Functionality of host driver related issues.
* RPU\_DEBUG\_IF = BIT(5),
  + To debug the Host Driver and RPU interface issue, this is for both CMD’s (host to RPU) and EVENTS (RPU to Host).
* RPU\_DEBUG\_UMACIF = BIT(6)
  + For debugging the interface issue between mac80211 and RPU Host Driver.
* RPU\_DEBUG\_RX = BIT(7),
  + For Debugging RX Data Path related issues.
  + Note: These debug’s can be huge.
* RPU\_DEBUG\_HAL = BIT(8),
  + For debugging host HAL to RPU HAL communication related issues.
  + Note: These debug’s can be huge.
* RPU\_DEBUG\_CRYPTO = BIT(9),
  + For Debugging Crypto related issues. Useful in debugging key programming issues.
* RPU\_DEBUG\_DUMP\_RX = BIT(10),
  + For Detail dumping of receive packets, useful in analysing receiving packets.
* RPU\_DEBUG\_DUMP\_HAL = BIT(11),
  + For Detail dumping of HAL Commands/Events, useful in debugging communications issue with RPU.
* RPU\_DEBUG\_TSMC = BIT(12),
  + To debug TSMC functionality issues, control path. For Data path use TX module.
* RPU\_DEBUG\_P2P = BIT(13),
  + To debug P2P functionality issues, control path. For Data path use TX module
* RPU\_DEBUG\_VIF = BIT(14),
  + To debug VIF functionality issues, control path.
* RPU\_DEBUG\_DUMP\_TX = BIT(15),
  + To dump content of TX packets payload, useful debugging data integrity issues.

When we issue the below command, from the output under the run time debug section these modules are listed in MODULE (Value) format, for all the modules we want to enable add all the values and save the sum.

**#cat /proc/rpu/params**

By default few essential debug features are enabled as shown below

* RPU\_DEBUG\_UMACIF
* RPU\_DEBUG\_CRYPTO
* RPU\_DEBUG\_MAIN

Once the value is derived we can configure it using the below 2 ways.

## Module Loading Time

We can enable the run time debug feature at module loading time, we can issue the below command:

**#insmod img\_wlan\_hostport.ko rpu\_debug=<value>**

Value: The sum saved above should be used.

**Eg1:** To enable Scan Related Debug’s. The command would be,

**# insmod img\_wlan\_hostport.ko rpu\_debug=2**

**Eg2:** Similarly, to enable multiple debug’s, we can use the above table to arrive at the value of “RPU\_debug” and then use as below

To enable below debugs, the command would be,

* CRYPTO Debug
* SCAN Debug

**#insmod img\_wlan\_hostport.ko rpu\_debug=1026**

## Run Time

We can enable the run time debug feature on the fly, by using the /proc interface of the host driver by issuing below command.

**#echo rpu\_debug=<total\_value> > /proc/rpu/params**

**Eg1:** To enable SCAN module debug.

**#echo rpu\_debug=2 > /proc/rpu/params**

**Eg2:** To enable CRYPTO and SCAN module debug.

**#echo rpu\_debug=1026 > /proc/rpu/params**

## Collecting run time logs

Please note that when this feature is enabled, the kernel logging might be heavy for some of the features and it may out run the kernel ring buffer for logging. So please use either of the approaches below to collect full logs

1. If prints are not so heavy, before the test case run below command, this will log all the kernel logs to console, so only console log should be sufficient.

**# dmesg -n 8**

1. If prints are heavy, we might lose control of the console, so stop all the logging, then run the test case, and for collecting the logs by issuing below command, this will collect the full logs from the kernel ring buffer.

Before starting the use case, disable console logs, they will still be logged in background.

**#dmesg -n 1**

After completion of use case, use this command to retrieve the background logs. Increase the buffer size to collect maximum possible logs.

**#dmesg -s 65535 -c**

# Post Processing

A Typical scenario for a problem found in the field is as follows

1. Problem reported from customer with the information obtained after running through sections #1 (configuration) and #2 (stats).
2. IMG will specify the list of debugs to be enabled. (Section #5), if needed.
3. Customer will enable the debugs and re-run the test case.
4. Customer will send updated data to IMG
5. IMG will do the post processing and debug the problem.

**Note:** Please note, the information captured above will only give preliminary insights in to the area/module of the problem, so that basic triage can be done. For deeper understanding/debug, access to RPU WLAN debug using IMG Codescape + Sysprobe (IMG JTAG) which is remotelyaccessible is necessary.