[HCP3] IMU BACKEND AND FRONTEND DRIVER GUIDE

HARMAN INTERNATIONAL

1. Revision History

Ver. No	Date	Revised Content	Description	Author
1.0	03-Oct-2023	All	First Version	Roopesh Kallumpurath

2. List of Terms

Term	Description
BE	Backend
CL43	Cluster Version 43 [ANDROID S Based]
CL45	Cluster Version 45 [ANDROID T Based]
FE	Frontend
HAL	Hardware Abstraction Layer
IIO	Industrial I/O subsystem
IMU	Inertial Measurement Unit
PV	Para-Virtualization
VIMU	Virtual IMU
VM	Virtual Machine

3. Contents

1. REVISION HISTORY 1

2. LIST OF TERMS 2

3. CONTENTS 3

4. PURPOSE & SCOPE 5

4.1 Purpose and Scope 5

5. ARCHITECTURE 6

5.1 General Description 6

5.2 IMU Architecture 6

5.3 Shared IMU Architecture 7
5.3.1 Shared IMU Block Diagram 7
5.3.2 Shared IMU Software Architecture 8
5.3.3 Shared IMU High Level Sequence Diagram 9

6. IMPLEMENTATION 10

6.1 IMU Driver 10
6.1.1 IMU Data Tapping 10
6.1.1.1 Files Modified 10
6.1.2 Disabled TIMER_BASED_BATCHING Configuration 11
6.1.2.1 Files Modified 11

6.2 IMU BE Driver 11
6.2.1 DTS Change 11
6.2.2 CONFIG Change 11
6.2.2.1 Kconfig and Makefile Changes 12
6.2.2.2 Source Files 12

6.3 IMU FE Driver 12
6.3.1.1 DTS Change 12
6.3.1.2 CONFIG Change 13
6.3.1.3 Kconfig and Makefile Changes 13
6.3.1.4 Source Files 13

7. APPLICATION NOTE 14

7.1 Pre-requisite: 14

7.2 Test Procedure: 14
7.2.1 Prepare Test Executable [domIVI] 14

7.2.2 Configure IMU driver [domSYS] 16
7.2.3 Run Test Applications 17

8. REFERENCES 18

4. Purpose & Scope

4.1 Purpose and Scope

This document provides HCP3 IMU backend and frontend driver Architecture, implementation and Application details.

5. Architecture

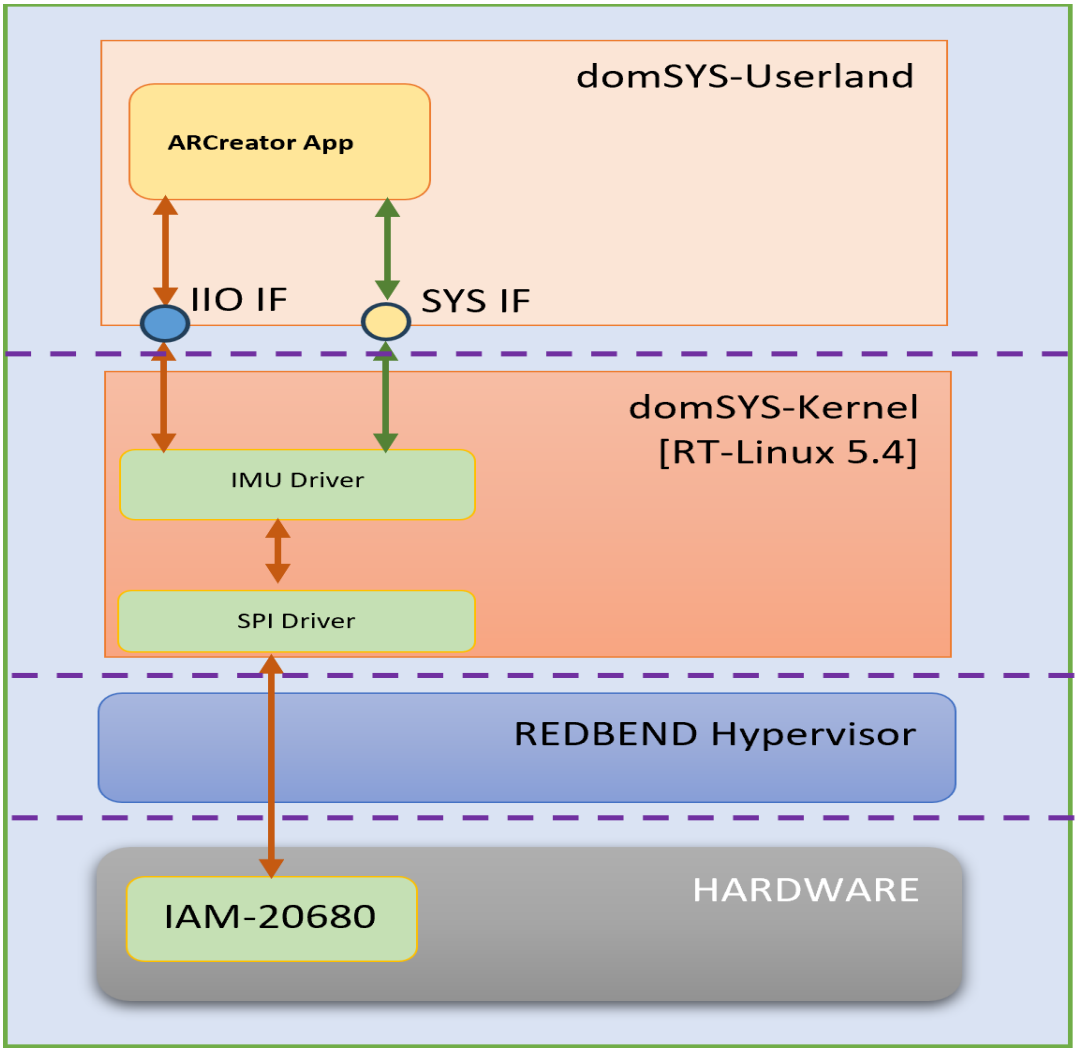
5.1 General Description

TDK InvenSense IAM-20680 chipset is used as the IMU sensor hardware in HCP3. The IAM-20680 is a 6-axis Motion Tracking device for Automotive applications that combines a 3-axis gyroscope and a 3-axis accelerometer in a small 3x3x0.75mm (16-pin LGA) package.

IMU driver provided by InvenSense TDK is based on the Industrial IO framework. Industrial I/O subsystem is a standard Linux interface to support devices with high data rate output. IMU InvenSense TDK driver is modified by Harman to adapt Audi specific data format and other specific requirements.

5.2 IMU Architecture

In IMU architecture [CL43], only the applications running on domSYS are using the IMU data. So, the IMU driver from InvenSense TDK is integrated into the SYS domain only.

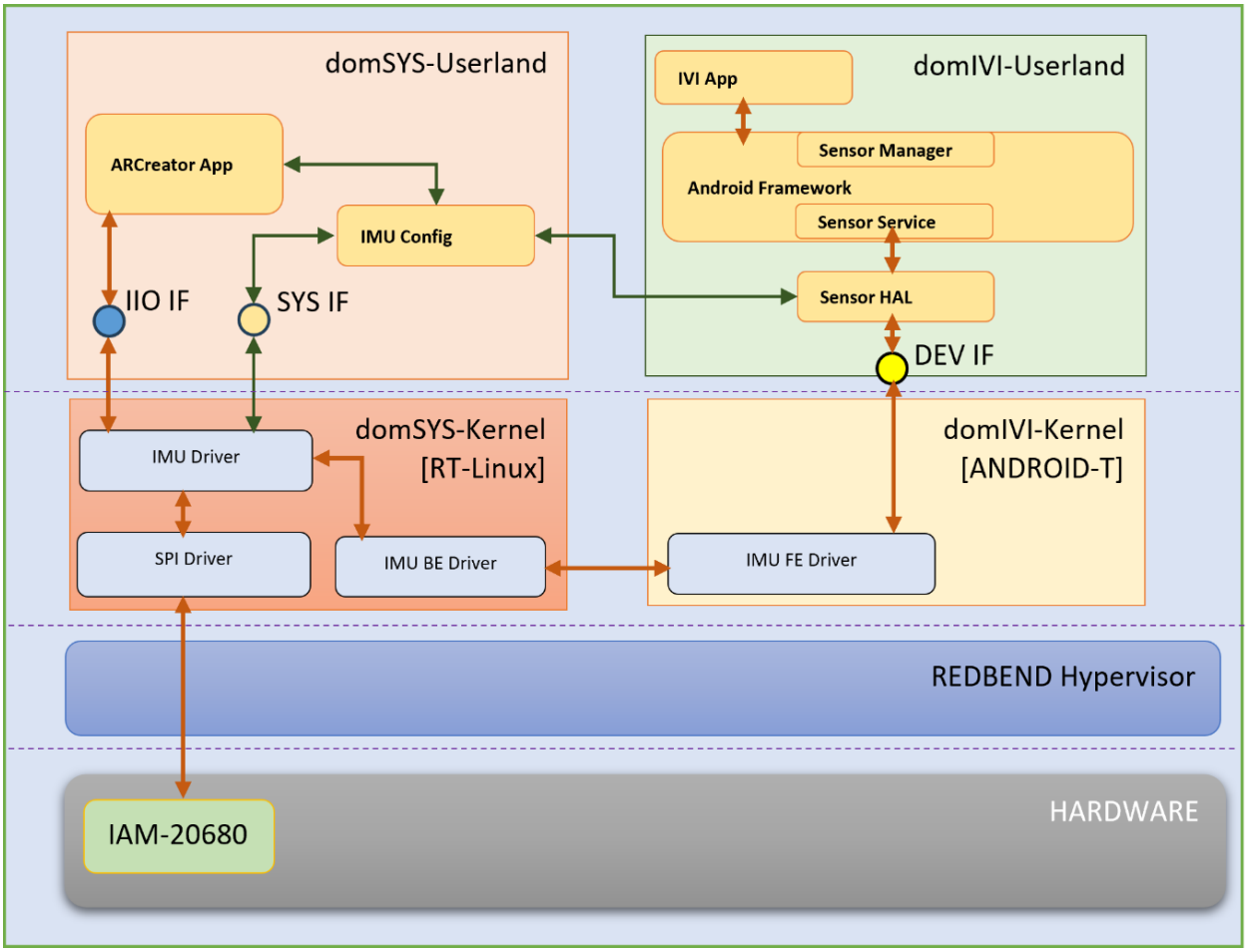


Please refer to document [IMU Driver Documentation V2.pdf] for more information.

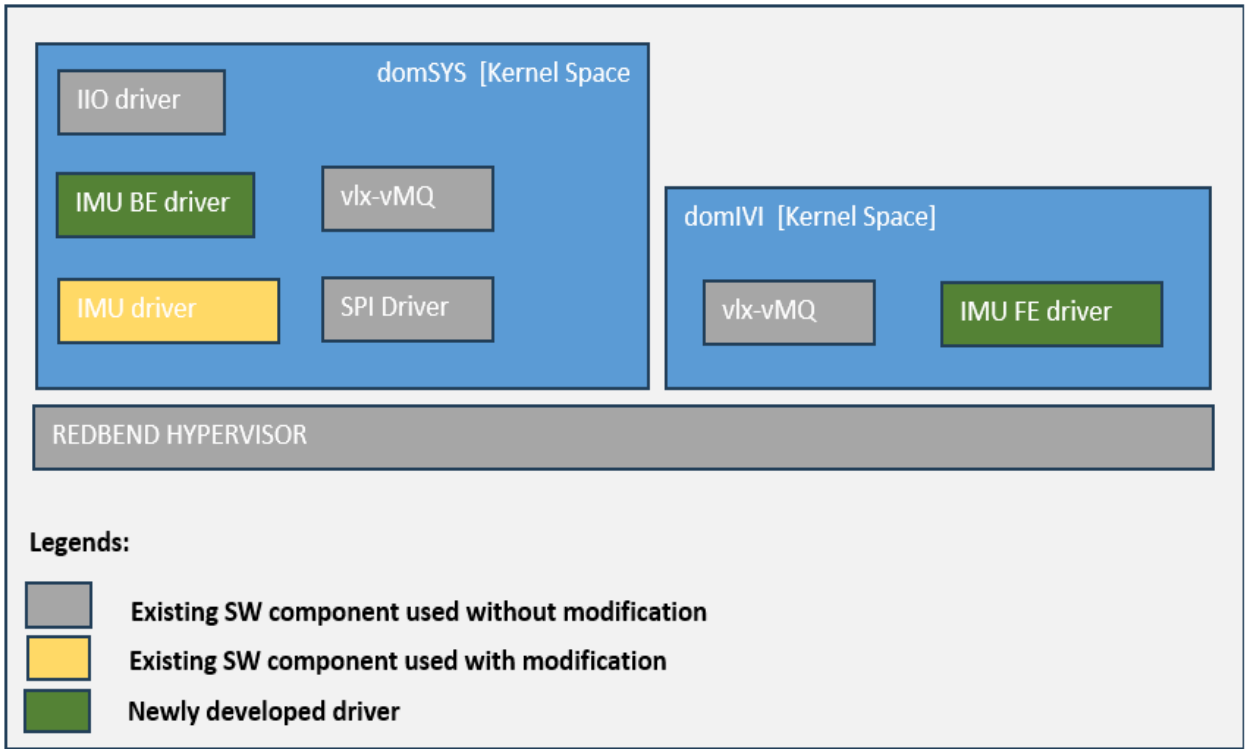
5.3 Shared IMU Architecture

In CL45, the applications running on both domSYS and domIVI need to access IMU data. So, the existing architecture is not sufficient for this requirement. To meet the new requirement, introduced a new shared IMU architecture.

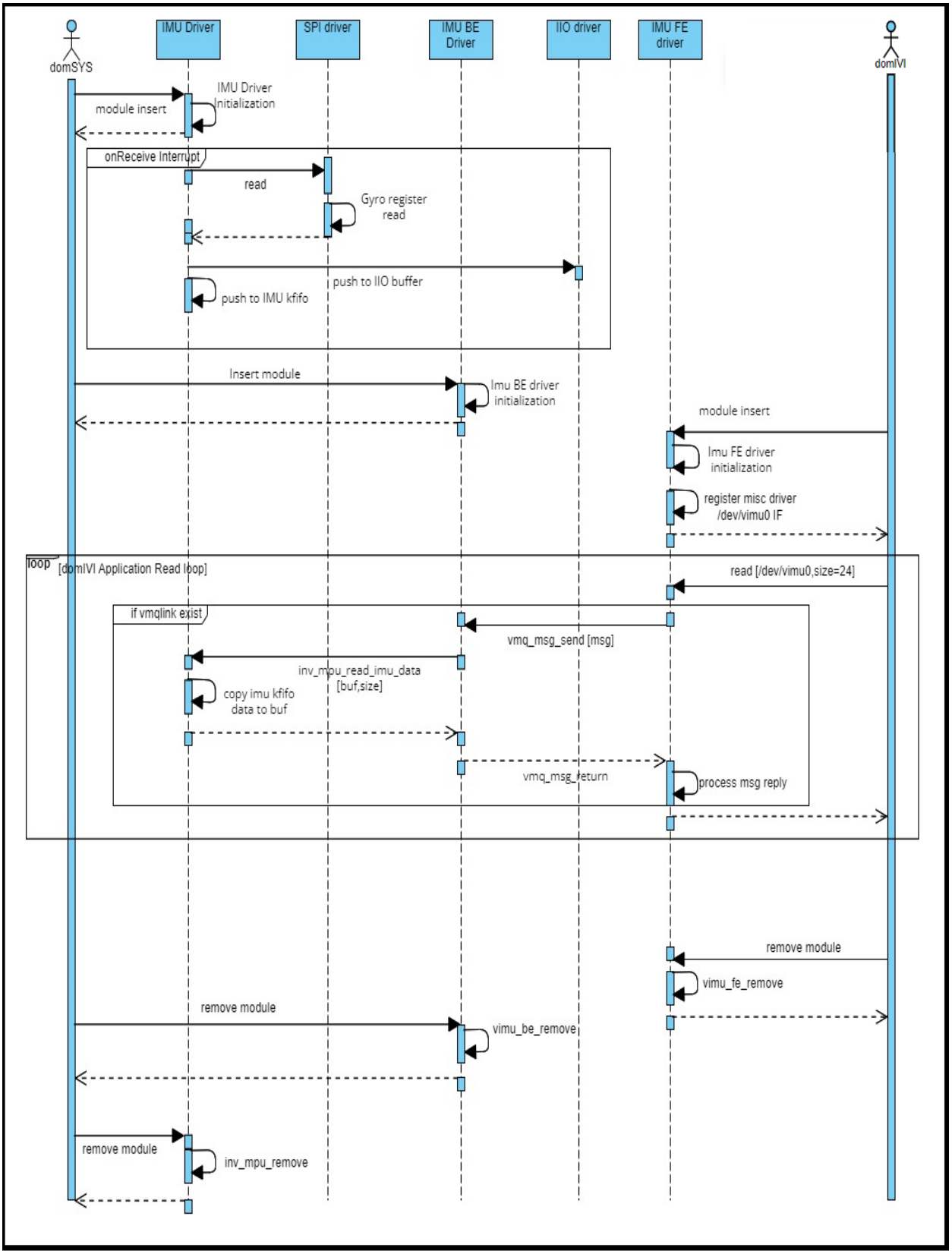
5.3.1 Shared IMU Block Diagram



5.3.2 Shared IMU Software Architecture



5.3.3 Shared IMU High Level Sequence Diagram



6. Implementation

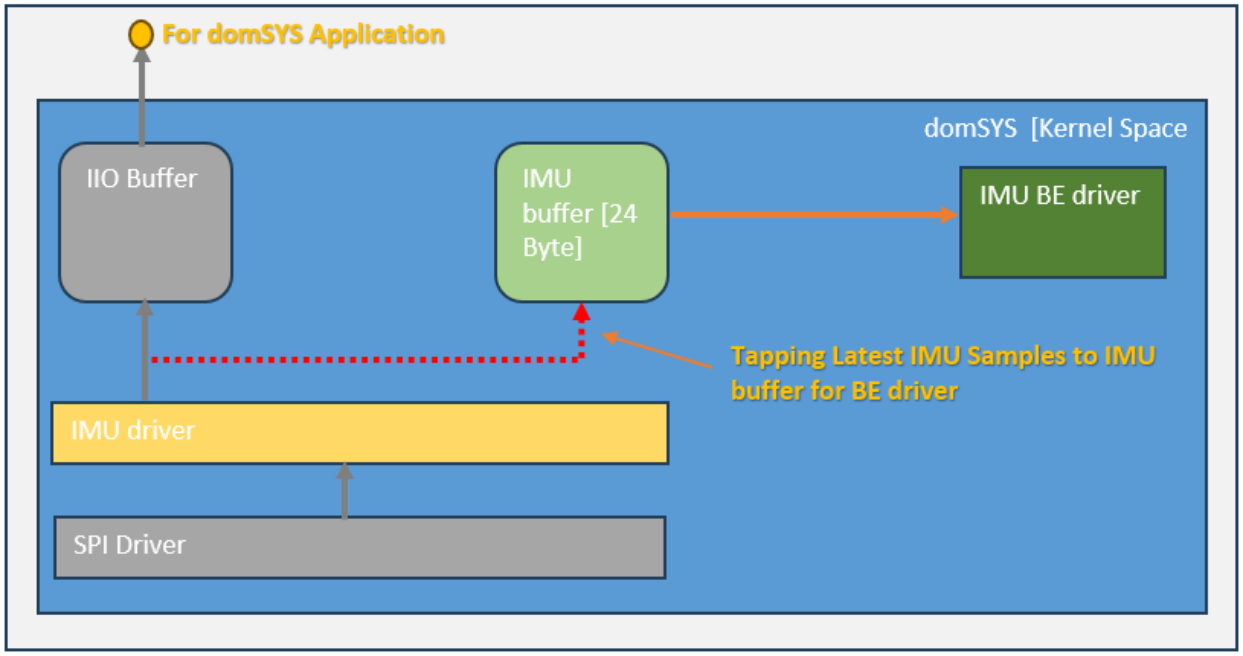
6.1 IMU Driver

The following sections explains the modifications done in the existing IMU driver to support IMU BE and IMU FE driver.

6.1.1 IMU Data Tapping

In the CL43 IMU architecture, the IMU data read from the IMU sensor is pushed to the IIO buffer, which is only available to the domSYS user space applications.

In the shared IMU architecture, pushing the latest IMU data to both the IIO buffer [for domSYS user space application] and a newly introduced local IMU buffer in the IMU driver [this tapped data in local IMU kfifo buffer acts as a source for IMU BE driver].



6.1.1.1 Files Modified

1. inv_mpu_iio.h
   - Added local buffer [imukf] in 'struct inv_mpu_state'
   - New API added [inv_mpu_read_imu_data]
2. inv_mpu_spi.c – Initialize the local buffer.
3. inv_mpu_parsing_20680.c – Tapped latest IMU sample in IMU local buffer [Inside function: inv_process_20680_data]
4. inv_mpu_ring.c – Added a new API definition 'inv_mpu_read_imu_data' to read the data from local IMU buffer [Used by IMU BE driver]

6.1.2 Disabled TIMER_BASED_BATCHING Configuration

If the 'TIMER_BASED_BATCHING' configuration is enabled, the IMU driver will process the data batch-wise. In batch processing, the driver will not read the individual samples from the IMU FIFO at every sampling interval. Instead, it will wait for the FIFO threshold to reach and initiate a read request when the data count in the FIFO register is greater than or equal to the FIFO threshold.

Example: If the sampling frequency is 4 Hz and the FIFO threshold is set as 12. Then the driver will initiate a read request at an interval of 3 seconds and read 12 samples in one shot.

In the shared IMU architecture, the IMU FE driver is designed in such a way that it will read one IMU data sample at a time (The application has to place a read request on every sampling period). So this design does not work with timer based batch process.

Example: If the IMU driver sampling frequency is set as 4Hz, then the applications running on the domIVI shall expect a new sample at every sampling interval (250ms). If timer batch processing is enabled, then the IMU driver cannot generate a new sample at every sampling interval. Due to this limitation 'TIMER_BASED_BATCHING' is disabled in the shared IMU architecture..

6.1.2.1 Files Modified

1. inv_mpu_iio_reg_20680.h – commented '#define TIMER_BASED_BATCHING'.
2. inv_mpu_iio.h – Fixed compilation error related to fifo_threshold.

6.2 IMU BE Driver

In the Shared IMU architecture, the IMU data needs to be shared between the domSYS and the domIVI. The IMU BE driver in the domSYS manages the data read request from the IMU FE driver. When a IMU data read request is received from the IMU FE driver through PV communication mechanism, the IMU BE driver processes this request, read the latest IMU data from the IMU driver and sends the data to the IMU FE driver.

The following sections explains the modifications done in the domSYS kernel to support IMU BE driver.

6.2.1 DTS Change

Added new platform node for IMU BE driver in the domSYS DTS file.

File: linux_sys-hcp3/exynosauto9-a994-linux-vm.dts

```
+       vimu_be: vimu_be {
+               compatible = "vl,vimu-be-vmq";
+               vl,vlink = "vmq_test";
+               status = "okay";
+       };
```

6.2.2 CONFIG Change

Enabled configuration for IMU BE driver in domSYS kernel configuration file.

File: configs/exynosauto9_hcp3_la_sys.fragment

```
+CONFIG_VLX_VIMU_BE=m
```

6.2.2.1 Kconfig and Makefile Changes

Added following changes in vlx Kconfig and Makefile.

File: drivers/vlx/Kconfig

```
+config VLX_VIMU_BE
+	tristate "Virtual IMU backend"
+	default n
+	depends on VMQ
+	depends on INV_MPU_IIO
+	select INV_MPU_IIO_IAM20680
+	help
+	  To compile this driver as a module, choose M here: the
+	  module will be working as a backend for IMU.
```

File: drivers/vlx/Makefile

```
+#VLX virtual IMU backend driver
+vimube-objs-y := vimu-be.o
+$(call vlx-vdriver, vimu-be, $(vimube-objs-y))
+
```

6.2.2.2 Source Files

Added the following new source files for IMU BE driver.

```
drivers/vlx/vimu-be.c
drivers/vlx/vimu-common.h
```

6.3 IMU FE Driver

The IMU FE driver manages the data read request from the domIVI user applications. When a data read request from the domIVI is received, the IMU FE driver communicates with the IMU BE driver for the latest IMU data. Once the IMU FE driver receives the latest data from the IMU BE driver, make it available to the domIVI user-space applications via '/dev/' interface.

The following sections explains the modifications done in the domIVI kernel to support IMU FE driver.

6.3.1.1 DTS Change

Added new platform node for IMU FE driver in the domIVI DTS file.

File: and_ivi-hcp3/exynosauto9-a994-android-vm.dts

```
+	vimu_fe: vimu_fe {
+		compatible = "vl,vimu-fe-vmq";
+		vl,vlink = "vmq_test";
+		status = "okay";
+	};
+
```

6.3.1.2 CONFIG Change

Enabled configuration for IMU FE driver in domIVI kernel configuration file.

File: arch/arm64/configs/exynosauto9_and_defconfig

```
+CONFIG_VLX_VIMU_FE=m
```

6.3.1.3 Kconfig and Makefile Changes

Added following changes in vlx Kconfig and Makefile.

File: drivers/vlx/Kconfig

```
+config VLX_VIMU_FE
+	tristate "Virtual IMU front-end"
+	default n
+	depends on VMQ
+	help
+	  To compile this driver as a module, choose M here: the
+	  module will be working as a front end for IMU.
+
```

File: drivers/vlx/Makefile

```
+#VLX virtual IMU front end driver
+vimufe-objs-y := vimu-fe.o
+$(call vlx-vdriver, vimu-fe, $(vimufe-objs-y))
+
```

6.3.1.4 Source Files

Added the following new source files for IMU FE driver.

```
drivers/vlx/vimu-fe.c
drivers/vlx/vimu-common.h
```

7. Application Note

This section explains the test procedure for IMU BE and FE drivers.

7.1 Pre-requisite:

- IMU module shall be inserted [domSYS]
- IMU BE module shall be inserted [domSYS].
- IMU FE shall be inserted [domIVI].

7.2 Test Procedure:

7.2.1 Prepare Test Executable [domIVI]

A. Create a directory *vimu* inside *Android* workspace.
B. Create a *Android.mk* file inside *vimu* directory and paste the following contents in it.

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE := vimutest
LOCAL_MODULE_TAGS := optional

LOCAL_CFLAGS += -Wno-missing-field-initializers
LOCAL_CFLAGS += -Wno-unused-parameter

LOCAL_SRC_FILES := \
        vimu_test.c
include $(BUILD_EXECUTABLE)
```

C. Create a source file named *vimu_test.c* inside *vimu* directory and write test code in it. A sample test code is provided below:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

static int DELAY_TABLE [9] =
{250000,125000,100000,20000,10000,5000,4000,2000,1000};

int main(int argc, char * argv[])
{
    int fd;
    int ret;
    char buf[24];
    char path[] = "/dev/vimu0";
```

```
char *sf;
int sf_index;
int delay_us;
int delta = 240; // delay to transfer 24 samples

if (argc != 2) {
        printf("Wrong sampling frequency Index. Using default index 4.\n");
        sf_index = 4;
}

else {
        sf = argv[1];
        sf_index = atoi(sf);
        if ((sf_index < 0) || (sf_index > 8))
        {
                printf("Wrong sampling frequency Index. Using default index 4.\n");
                sf_index = 4;
        }
}

delay_us = DELAY_TABLE[sf_index];

fd = open(path, O_RDONLY);

if (fd == -1) {
   printf ("File open failed!\n");
   return -1;
}

while (1) {

        ret = read(fd, buf, 24);
        if (ret == 24) {
                printf("%02X %02X %02X %02X %02X %02X %02X %02X ",
                buf[0],buf[1],buf[2],buf[3],buf[4],buf[5],buf[6],buf[7]);
                printf("%02X %02X %02X %02X %02X %02X %02X %02X ",
                buf[8],buf[9],buf[10],buf[11],buf[12],buf[13],buf[14],buf[15]);
                printf("%02X %02X %02X %02X %02X %02X %02X %02X\n",
                buf[16],buf[17],buf[18],buf[19],buf[20],buf[21],buf[22],buf[23]);
        }

        usleep (delay_us- delta);
}

if (fd)
        close (fd);
```

```
    return 0;

}
```

D. Source the build environment script and lunch configuration

```
/home/user/Android/vimu$ source ../build/envsetup.sh
/home/user/Android/vimu$ lunch hcp3_e3_1_2-userdebug
```

E. Build using 'mm' command.

```
/home/user/Android/vimu$ mm
```

F. The output file will be generated inside 'Android/out/target/product/hcp3_vm/system/bin/' directory. Copy the executable to Android path '/misc'.

7.2.2 Configure IMU driver [domSYS]

1. Verify the IMU and IMU BE drivers are successfully inserted. Use *lsmod* command to verify the module insertion status. The following modules shall be listed in the *lsmod* output.

```
inv-mpu-iio
vimu-be-module
```

2. Configure the IMU driver as shown below:

```
cd /sys/devices/platform/103a0000.spi/spi_master/spi5/spi5.0/iio:device0
echo 32768 > buffer/length
echo 3 > in_anglvel_scale
echo 3 > in_accel_scale
echo 0 > sampling_frequency
echo 0 > buffer/enable
echo 1 > scan_elements/in_anglvel_y_en
echo 1 > scan_elements/in_anglvel_x_en
echo 1 > scan_elements/in_anglvel_z_en
echo 1 > scan_elements/in_accel_x_en
echo 1 > scan_elements/in_accel_y_en
echo 1 > scan_elements/in_accel_z_en
echo 1 > scan_elements/in_temp_en
echo 1 > scan_elements/in_timestamp_en
echo 1 > buffer/enable
echo 1 > in_sensor_enable
```

The sampling frequencies supported by IMU driver is given below:

Index	Frequency in Hz
0	4
1	8
2	10
3	50
4	100
5	200
6	250
7	500
8	1000

In the above example, selected index 0. So, it will set 4 Hz as the sampling frequency.

7.2.3 Run Test Applications

1. Start capturing the IMU samples in domSYS using the below commands.

```
cat /dev/iio\:device0 |hexdump -v -e '24/1 " %02X" "\n"'  > /tmp/
imu_sys_samples_4hz.txt
```

2. Start capturing the IMU samples in domIVI using below commands.

```
/misc/vimutest 0 > /misc/imu_ivi_samples_4hz.txt
```

   Note: ‘0’ is the sampling frequency index command-line parameter in the above example. Select the command line parameter according to the sampling frequency index.

3. Run the test for adequate time frame and stop the applications running on SYS and IVI. Then compare the contents of *‘/tmp/ imu_sys_samples_4hz.txt’* from SYS and *‘/misc/imu_ivi_samples_4hz.txt’* from IVI. The content shall be matching.

8. References

1. CDD_AUDI_HCP3_Kernel_GYRO.pdf
2. IMU Driver Documentation V2.pdf
3. Exynosauto9HypervisorPVDriverGuide_Miscellaneous_ALL_REV 1.00.pdf
4. https://confluence.harman.com/confluence/display/AUDIHCP3/Shared+IMU+-+Architecture
5. https://confluence.harman.com/confluence/pages/viewpage.action?spaceKey=ELINA&title=IMU+Bring+up+on+Android+T+Sys+Side