Linux Performance Test Bench

Reference Guide

Literature Number: SPRUFQ7
JULY 2008

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Read This First

About This Manual

This document describes the overall architecture of pspTest tool. This document intends to help you to modify the code for any customization. Linux Performance Test Bench is part of the pspTest tool and can be found at ./psp_test_bench/performanceTest.

Intended Audience

This document is intended for the users of LSP package who might want to know the architecture of Linux Performance Test Bench (LPTB), design / procedures followed for measuring the performance of various device drivers using the LPTB and procedure to follow for adding new test cases to the LPTB.

This document assumes that the user has hands on experience with the Linux platform and some knowledge regarding the LSP device drivers for which performance parameters are being measured.

How to Use This Manual

This document includes the following chapters:

- **Chapter 1 – Introduction ,**gives an overview of Linux Performance Test bench
- **Chapter 2 - pspTest Tool Directory Structure**, describes the files and directory structure of the installed pspTest package in the system.
- **Chapter 3 - pspTest Architecture and Utilities**, describes the overall architecture and utilities of the user level tests and kernel level tests supported by the pspTest tool.
- **Chapter 4 - Test Modules and Flowcharts**, provides the test modules and flow charts for all the modules.

Terms and Abbreviations

The following terms and abbreviations are used in this document.

Term/Abbreviation	Description
DUT	Device Under Test

Term/Abbreviation	Description
API	Application Programming Interface
IO	Input/Output
IOCTL	Input Output ConTroL
DMA	Direct Memory Access
EDMA	Enhanced Direct Memory Access
QDMA	Quick Direct Memory Access
ATA	Advanced Technology Attachment
MMCSD	MultiMedia Card / San Disk
CCV	Chrominance ConVersion
USB	Universal Synchronous Bus
fps	Frames Per Second
NTSC	National Television System Committee
PAL	Phase Alternating Line
TV	TeleVision
VDCE	Video Data Conversion Engine
VPIF	Video Processing InterFace
VPSS	Video Processing Sub System

If You Need Assistance

For any assistance, send a mail to dsppsp_val@list.ti.com.

Text Conventions

The following conventions are used in this document:

- Text inside back-quotes (‘) represents pseudo-code.
- Program source code, function and macro names, parameters, and command line commands are shown in a `mono-spaced` font.

Trademarks

Code Composer Studio, the DAVINCI Logo, DAVINCI, DSP/BIOS, eXpressDSP, TMS320, TMS320C64x, TMS320C6000, TMS320DM644x, and TMS320C64x+ are trademarks of Texas Instruments.

This page is intentionally left blank

Contents

Figures

Tables

Revision History

Date	Author	Comments	Version
March 28, 2008	Asha B N	Initial Draft	0.1.0
May 26, 2008	Surendra Puduru	Updated as per the review comments	0.1.1
May 26, 2008	Som	Updated the user guide for directory structure and naming convention changes	0.1.2
May 26, 2008	Prachi	Updated the user guide video changes	0.2.0
June 26, 2008	Surendra Puduru	Updated with the details for CPU Occupancy and required changes to flow charts	0.2.1
July 2, 2008	Som	Updated CPU Occupancy	0.2.2

Chapter 1

Introduction

This chapter describes the services, features, limitations, and requirements of the Linux Performance Test Bench.

1.1 Overview

Linux Performance Test Bench supports benchmarking of various Linux device drivers supplied as part of the Linux Support Packages (LSP) for TI platforms. The current package support throughput and CPU Occupancy measurements for the device driver IO operations. This product can be scaled up to add support for new drivers, new platforms, and additional performance parameters.

1.1.1 Supported Services

Linux Performance Test Bench provides the code to get performance parameters for the following device drivers:

- Video
- VDCE
- Audio
- EDMA
- File System
- I2C
- SPI
- VLYNQ
- USB ISO Video

1.1.2 Supported Features

Following are the supported features:

- Linux Performance Test Bench supports throughput measurement for both User Level and Kernel Level device drivers.
- Linux Performance Test Bench supports CPU load measurement for User Level device drivers.
- Using the scripts available in the package for all the device drivers, throughput can be measured with minimal manual effort.
- Using the command line, user can get throughput of all the user level device drivers for various input parameters like different buffer sizes, sampling rates etc.
- Common methods are used for buffer allocation, time measurement for performance calculations.

1.2 Limitations

Following are the limitations:

- Boundary checking for Input parameters given through command line is not taken care. So user should give the input parameters accordingly.
- CPU load measurement for kernel level modules and memory requirements while measuring throughput will be implemented in later phase.
- Directory structure is prone to changes when adding support for new platforms.

1.3 Basic Hardware and Software Requirements

1.3.1 Hardware Requirements

Refer to the respective performance test bench user guides for the Hardware required for using the Linux Performance Test Bench.

1.3.2 Software Requirements

Linux Support Package for TI Platforms which includes

- Device drivers required for DSP
- Source for U-Boot

This page is intentionally left blank

Chapter 2

pspTest Tool Directory Structure

This chapter describes the files and directory structure of the installed pspTest package in the system. Refer to the respective performance test bench user guides for the installation procedure of pspTest package.

2.1 Top Level Directory Structure

Figure 2-1. shows the top level directory structure after installing the pspTest tool.

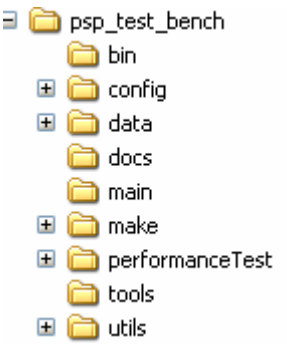


Figure 2-1. Directory Structure

Following are the directories in this package.

- /bin – This contains Executables of pspTest tool
- /config – This contains configuration script for configuring pspTest tool and generated configuration files
- /data – This contains input files used by the performance tests
- /docs – This contains test bench User guides, Release Notes, and reference guide documents
- /main – This contains code for Initializations
- /make – This contains make files for target and host compilation
- /performanceTest – This contains source code for performance measurements of various test modules for performance vectors such as throughput etc
- /tools -- This contains performance tools for modules which are not supported by performance test bench
- /utils – This contains utilities such as timer, CPU Load, tokenizer, and buffer manager used by the user level test modules and kperftimer used by kernel level module

2.2 Performance Test Directory Structure

Figure 2-2 shows the performance test directory structure for test modules.

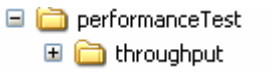


Figure 2-2.Performance Test Directory Structure

2.3 Test Modules

This section provides the details of the directory structure for test modules.

2.3.1 Throughput Directory Structure

Figure 2-3 shows the throughput directory structure for test modules.

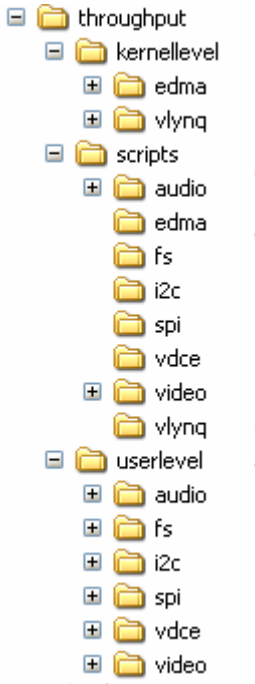


Figure 2-3. Throughput Directory Structure

- /kernellevel - This contains tests for modules that can be tested only at the kernel level.
  - /edma - This contains source code for EDMA and QDMA ASync/ABSync throughputs
  - /vlynq - This contains source code for VLYNQ read and write throughput using EDMA and CPU
- /scripts - This contains individual module scripts for executing the performance test bench.
  - /audio – This contains audio scripts for ALSA and OSS

- /edma – This contains scripts for EDMA
- /fs– This contains scripts for NAND, NOR, USB-MSCHOST and ATA
- /i2c – This contains scripts for I2C
- /spi – This contains scripts for SPI
- /vdce – This contains scripts for VDCE
- /video – This contains video scripts for FBDEV, V4L2 and USB-Video
- /vlynq – This contains scripts for VLYNQ

- /userlevel - This contains tests for modules that can be tested at the user level.
  - /audio - This contains source code for read and write throughput of audio.
  - /fs - This contains source code for read and write throughput for file system.
  - /i2c - This contains source code for read and write throughput for I2C.
  - /spi - This contains source code for read and write throughput for SPI.
  - /vdce - This contains source code for Resize, chrominance conversion (420 to 422 and 422 to 420), Blending, Edge Padding, and Range mapping throughput
  - /video - This contains source code for Capture and Display, and throughput for VPIF and USBISO video capture.

2.4 Make directory structure

Figure 2-4 shows the directory structure for make module

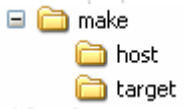


Figure 2-4. Make Directory Structure

2.5 Config directory structure

Figure 2-5 shows the directory structure for config module

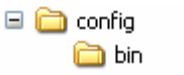


Figure 2-5. Config Directory Structure

- /config – This contains generated configuration files CURRENTCFG.MK and config.h
- /bin – This contains configuration script for configuring pspTest tool

Chapter 3

pspTest Architecture and Utilities

This chapter describes the overall architecture and utilities of the user level tests and kernel level tests supported by the pspTest tool.

3.1 pspTest Architecture for User Level Tests and Utilities

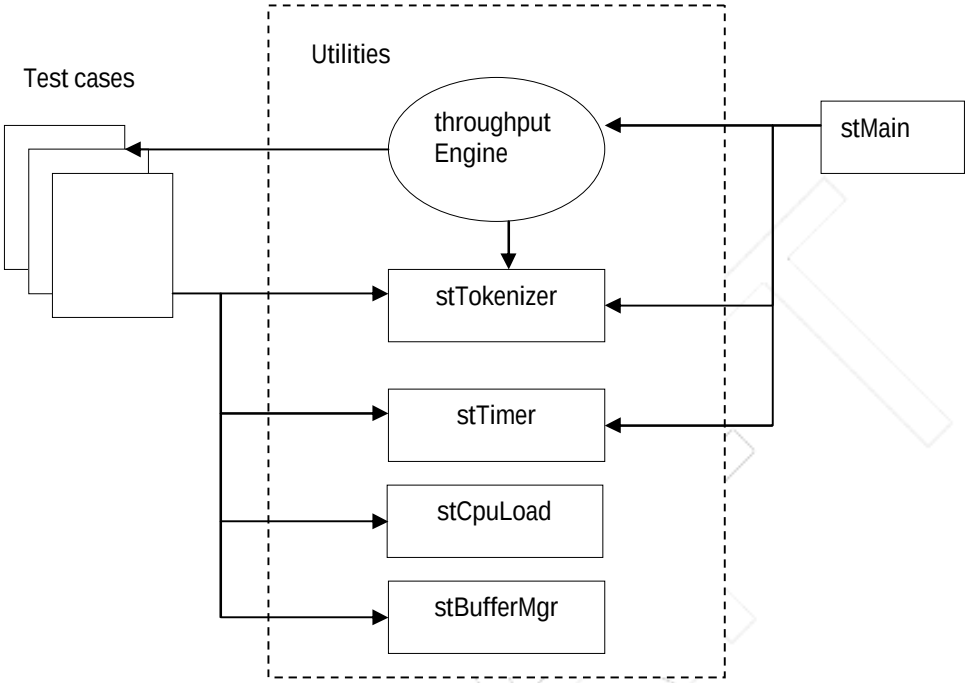


Figure 3-1. pspTest Framework

3.1.1 stMain

The stMain.c file invokes the following functions:

1. `initTimerModule` - Initializes the Timers used for measuring the performance. This function is defined in stTimer.c utility.
2. `getNextTokenString` - Gets the first parameter from the command line arguments. This function is defined in stTokenizer.c utility.
3. `handleThroughputTest` - Gets the command parameter from the command line arguments and invokes the respective function to execute the command. This function is defined in throughputEngine.c utility.

3.1.2 pspTest User Level Utilities

This section describes the pspTest user level utilities:

3.1.2.1 throughputEngine.c

- `handleThroughputTest ()` - Gets the command parameter from the command line arguments and invokes the respective function to execute the command.

3.1.2.2 stTokenizer.c

- `getNextTokenString()` - This function returns the next token string on the top and decrements `numArgs`.
- `getNextTokenInt()` - This function converts and returns the next token on the top into a integer and decrements `numArgs`.

3.1.2.3 stTimer.c

- `startTimer ()` - Gets the current time using `gettimeofday()` and sets the same to the argument passed.
- `stopTimer ()`- Gets the current time using gettimeofday() and returns the time elapsed between the current time and the time passed as argument in microseconds.

3.1.2.4 stCpuLoad.c

- `startCpuLoadMeasurement ()` - Gets the current CPU status using proc/stat and updates the same to the argument passed.
- `stopCpuLoadMeasurement ()`-Gets the current CPU status using proc/stat and returns the CPU Load using the time spend in different CPU states between the current CPU status and the CPU status passed as argument in jiffies. For more details refer Appendix A

3.1.2.5 stBufferMgr.c

- `stAllocateBuffer()` - Allocates the memory for the requested size.
- `stFreeBuffer()` - Free the memory allocated to the buffer pointer passed to it.

3.2 pspTest Kernel Level Tests and Utilities

Tests for EDMA and VLYNQ drivers are executed at kernel level and the tests are built as Linux Kernel Modules.

3.2.1 kStTimer.c

- `start_Timer ()` - This function is called to indicate the start of timing.
- `stop_Timer ()` - This function is called to indicate the end of timing and returns the time lapsed in microseconds.

This page is intentionally left blank

Chapter 4

Test Modules and Flowcharts

This chapter provides the test modules and flow charts for all the modules.

4.1 File IO Performance Tests

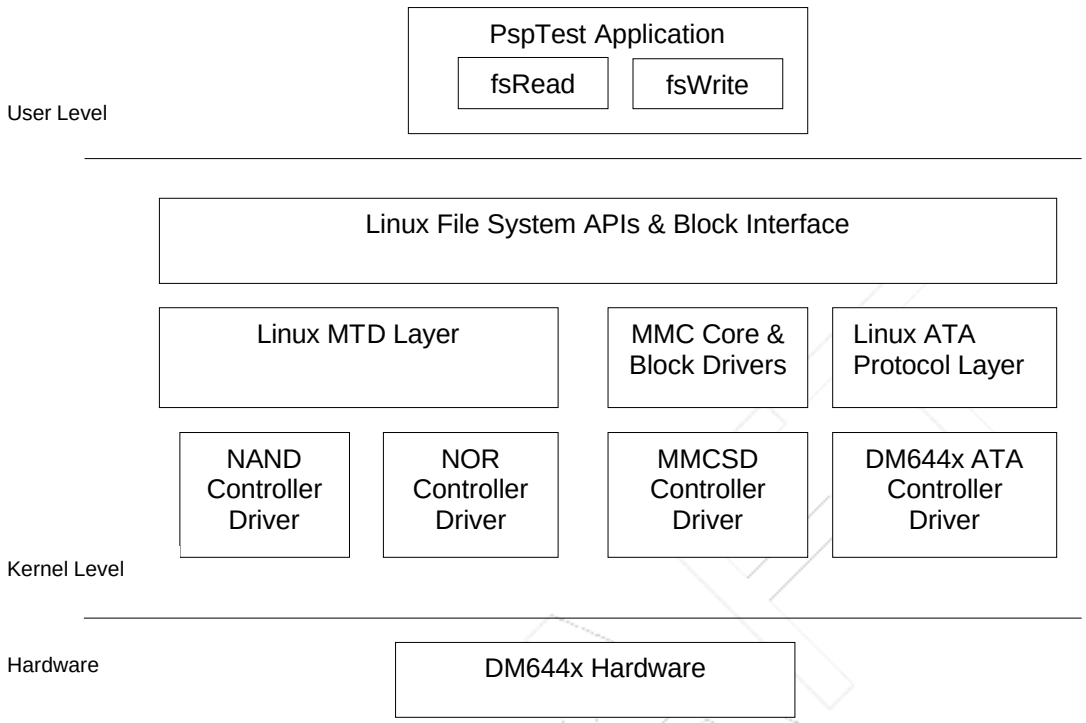


Figure 4-1. File IO Flow Diagram

4.1.1 File Read Performance

The read performances for storage drivers such as ATA, NAND, NOR, MMCSD, and USB is measured as time taken in seconds for reading a file of size 1MB.

The pspTest tool allows you to configure the buffer size used to read a file of any size depending on the maximum storage available on the device. Throughput can be measured either by mounting the device using a file system or by sector/raw access.

Figure 4-2 shows the flow chart for the function `throughputFSRead()`.

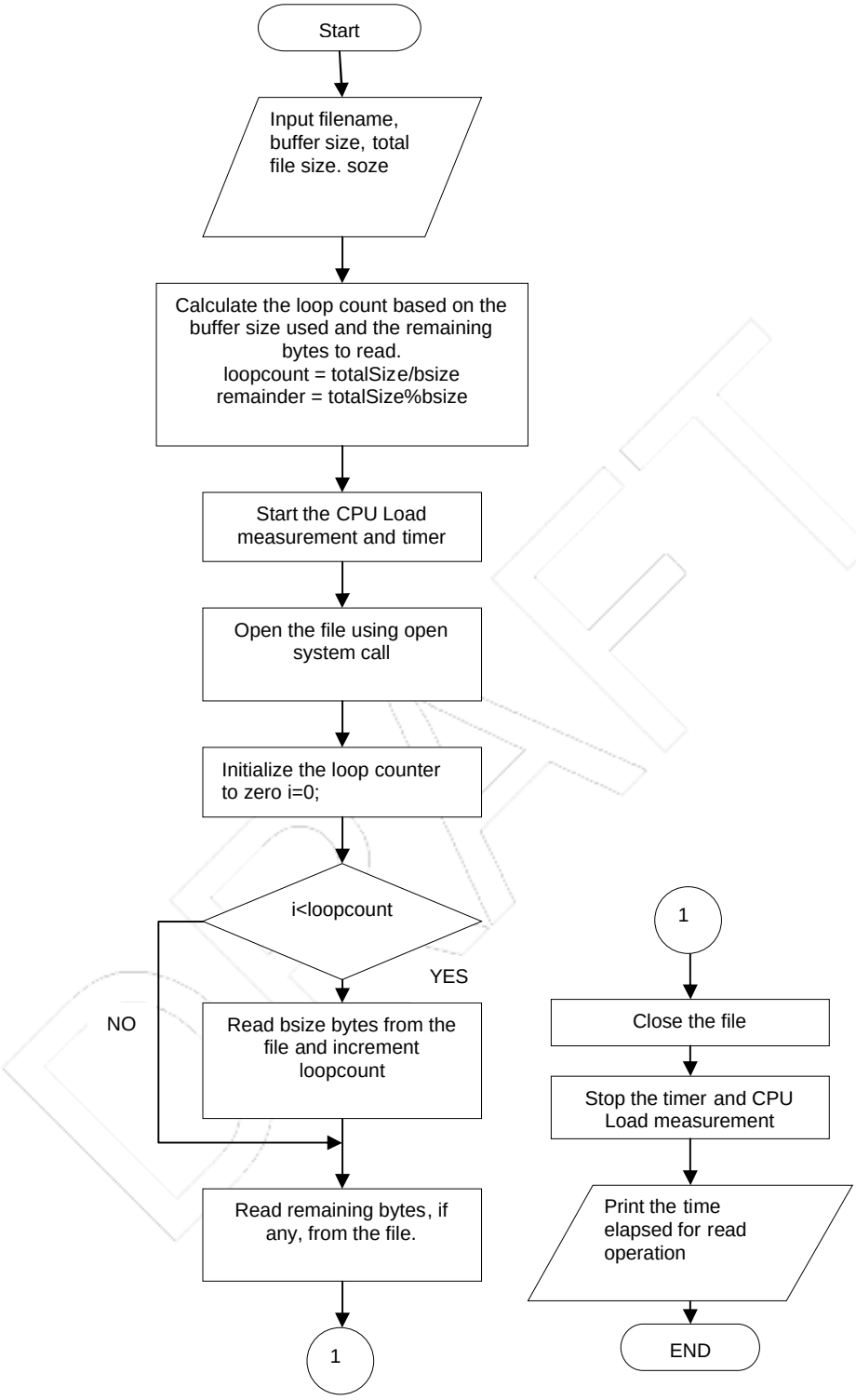


Figure 4-2. Flow Chart of File Read Performance

4.1.2 File Write Performance

The write performances for storage drivers such as ATA, NAND, NOR, MMCSD, and USB is measured as time taken in seconds for writing a file of size 1MB.

The pspTest tool allows you to configure the buffer size used to write a file of any size depending on the maximum storage available on the device. Throughput can be measured either by mounting the device using an appropriate file system or by sector/raw access.

Figure 4-3 shows the flow chart for the function `throughputFSWrite()`.

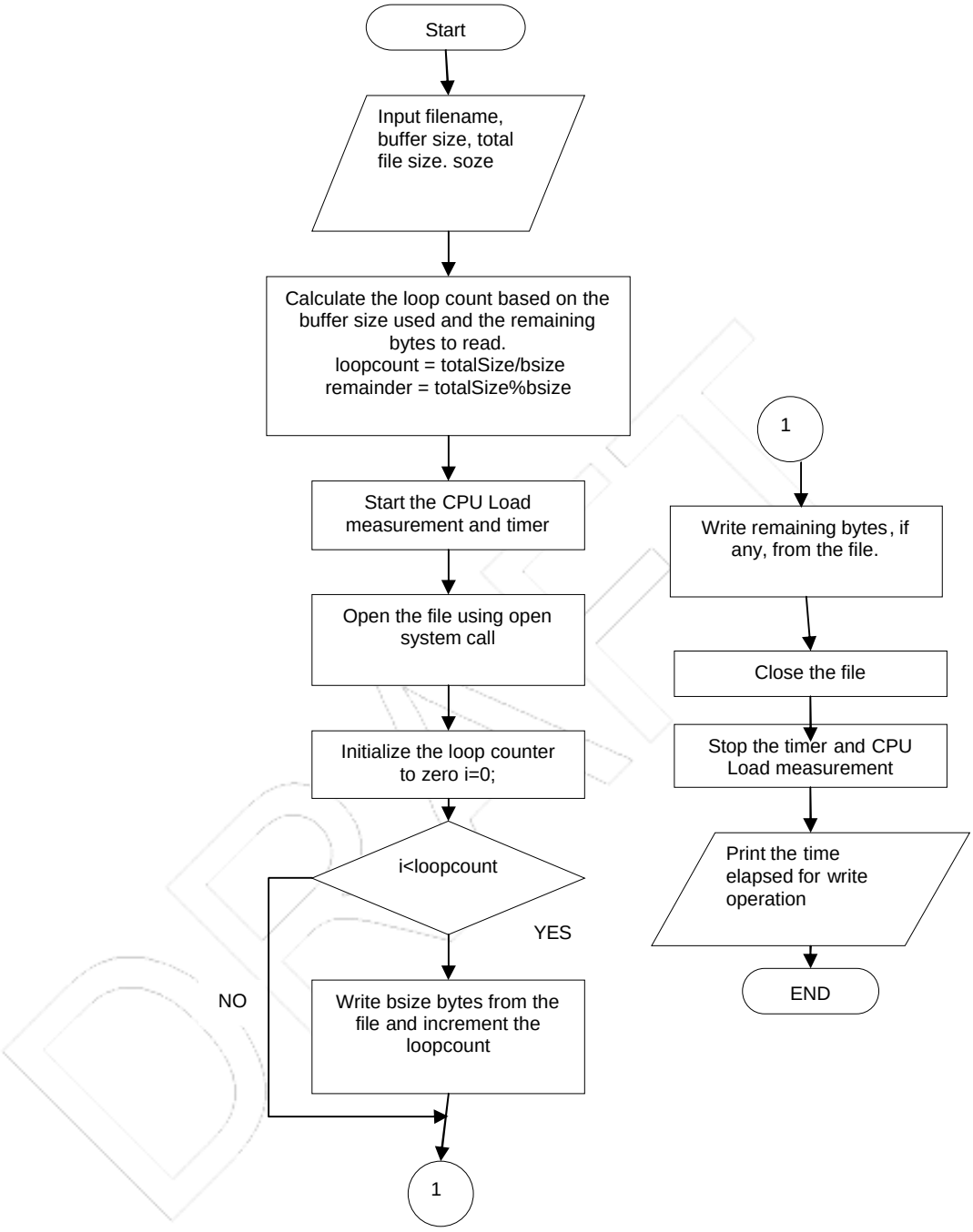


Figure 4-3. Flow Chart of File Write Performance

4.2 Audio Record and Playback Performance Tests

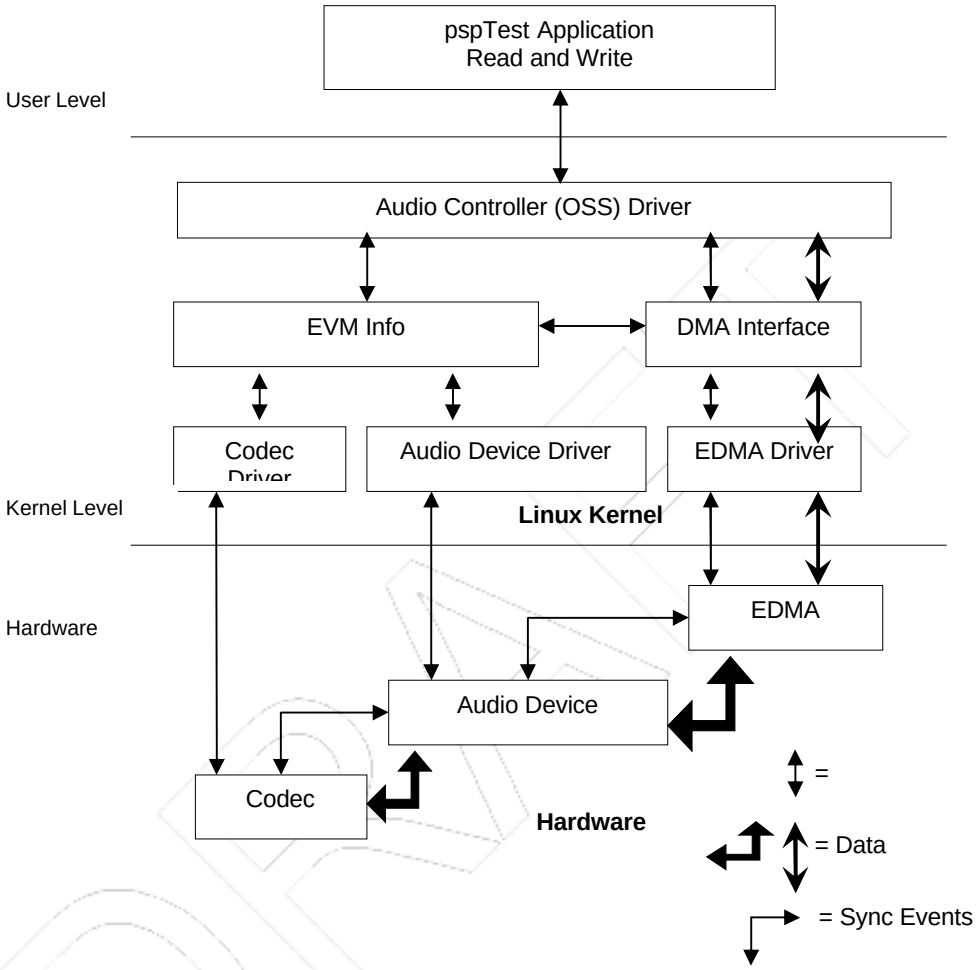


Figure 4-4. Audio Record and Playback IO Flow Diagram

4.2.1 Audio Read Performance

The read performance for Audio and USB ISO Audio drivers is measured as time taken in seconds to record audio data of size 5MB.

Figure 4-5 shows the flow chart for the function `throughputAudioRead()`.

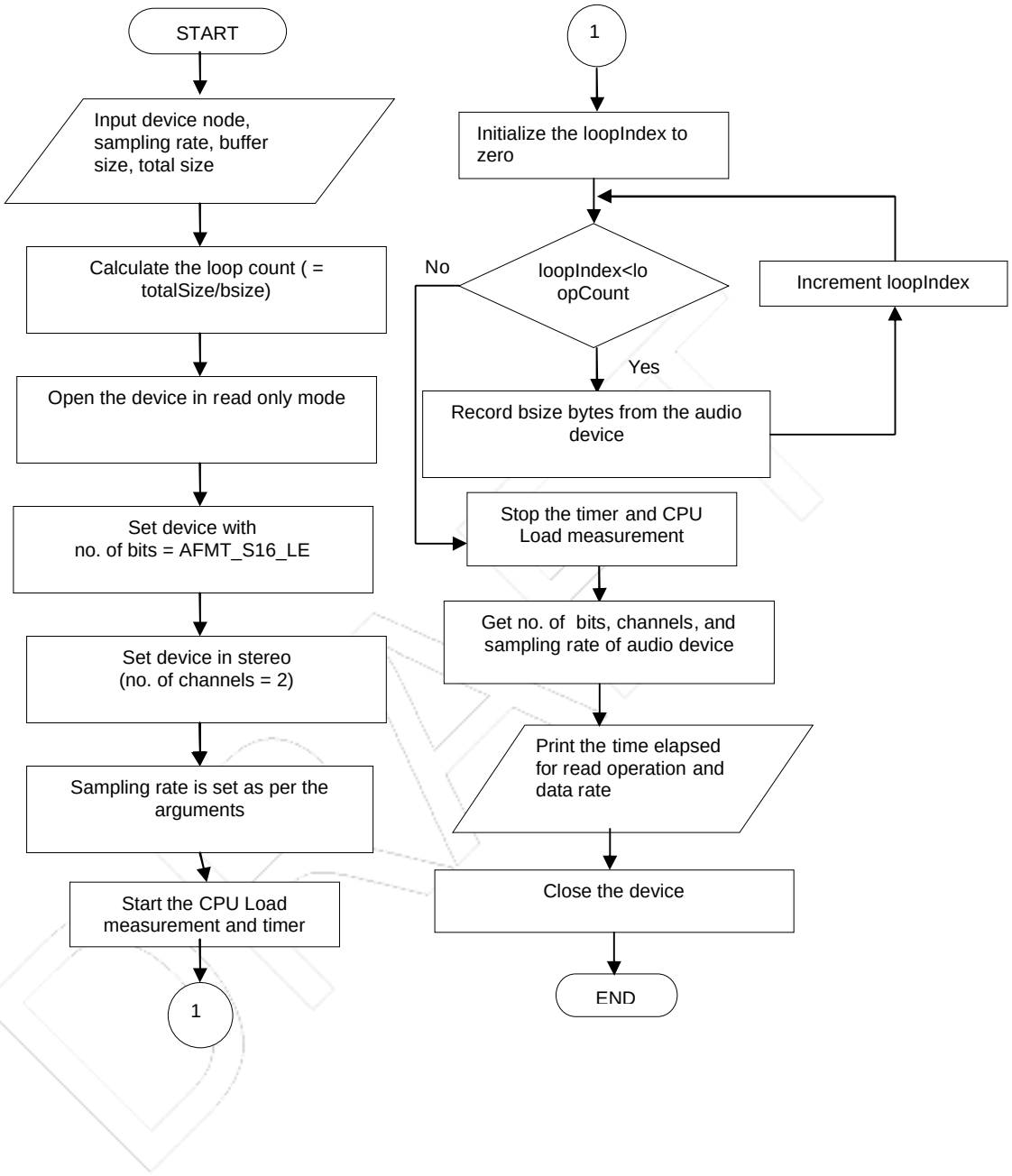


Figure 4-5. Flow Chart of Audio Read Performance

4.2.2 Audio Write Performance

The write performance for Audio and USB ISO Audio drivers is measured as time taken in seconds for playing back audio data of size 5MB.

Figure 4-6 shows the flow chart for the function `throughputAudioWrite()`.

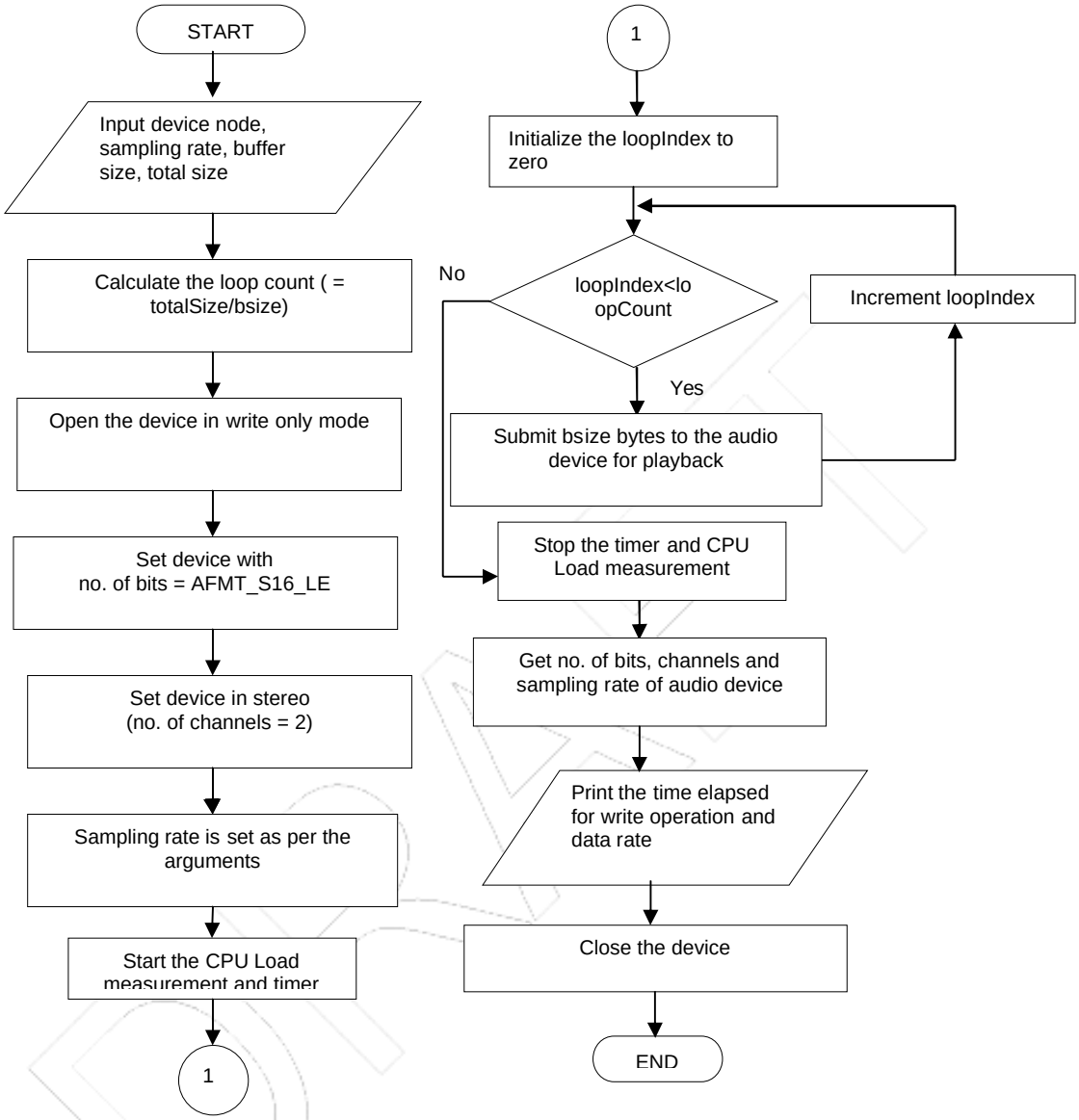


Figure 4-6. Flow chart of Audio Write Performance

4.2.3 Audio Read with Write to a File Performance

The read performance for Audio and USB ISO Audio drivers is measured as time taken in seconds for recording audio data of size 5MB and writing to a file.

Figure 4-7 shows the flow chart for the function `throughputAudioReadToFile()`.

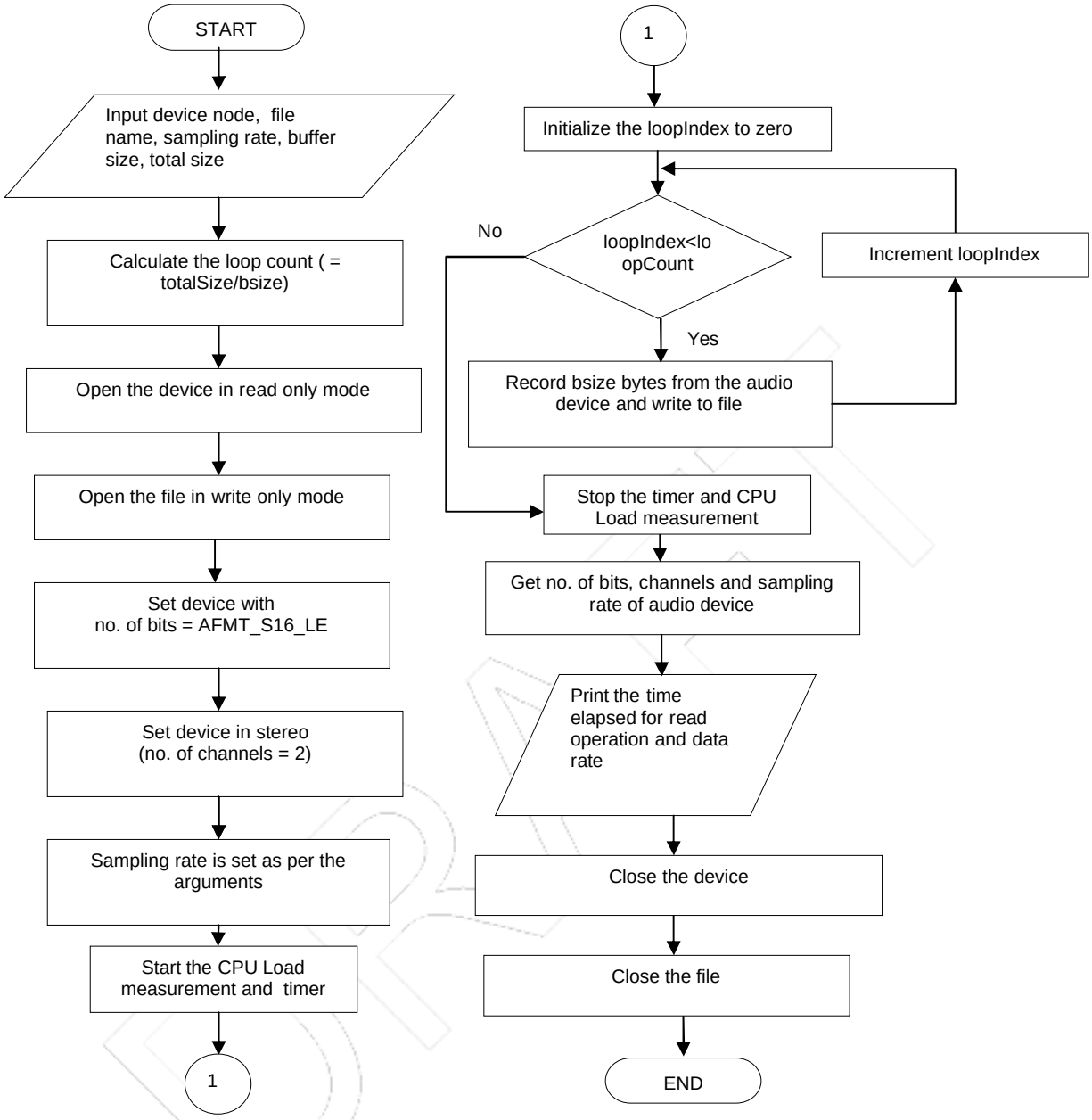


Figure 4-7. Flow chart of Audio Read with Write to a File Performance

4.2.4 Read from File and Audio Write Performance

The write performance for Audio and USB ISO Audio drivers is measured as time taken in seconds for recording audio data of size 5MB.

Figure 4-8 shows the flow chart for the function `throughputAudioWriteFromFile ()`.

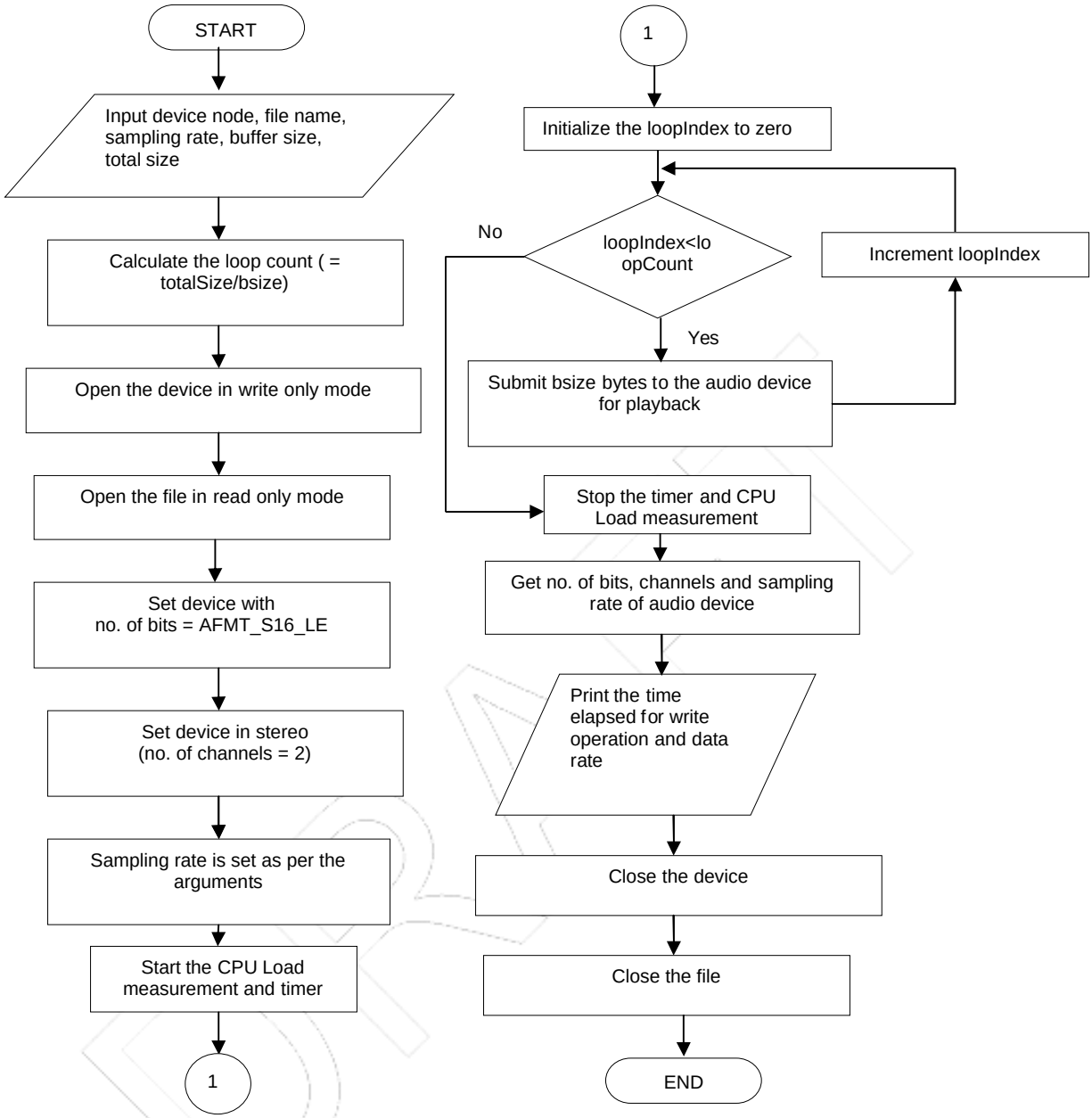


Figure 4-8. Flow chart of Read from File and Audio Write Performance

4.3 USB ISO Video Capture Performance Tests

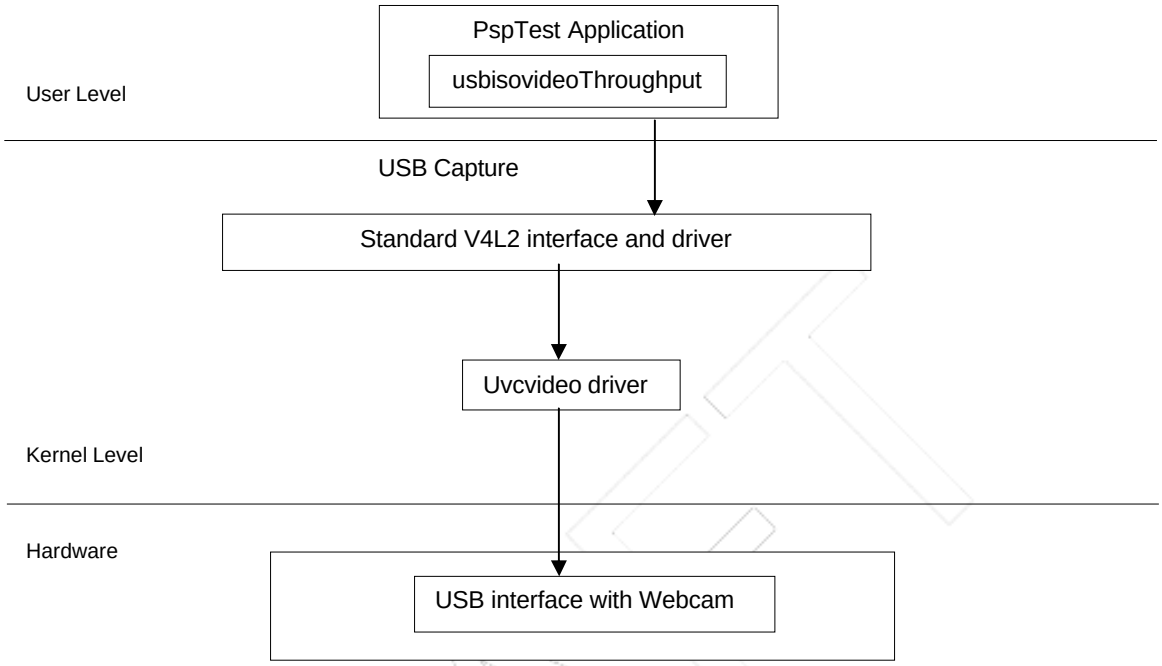


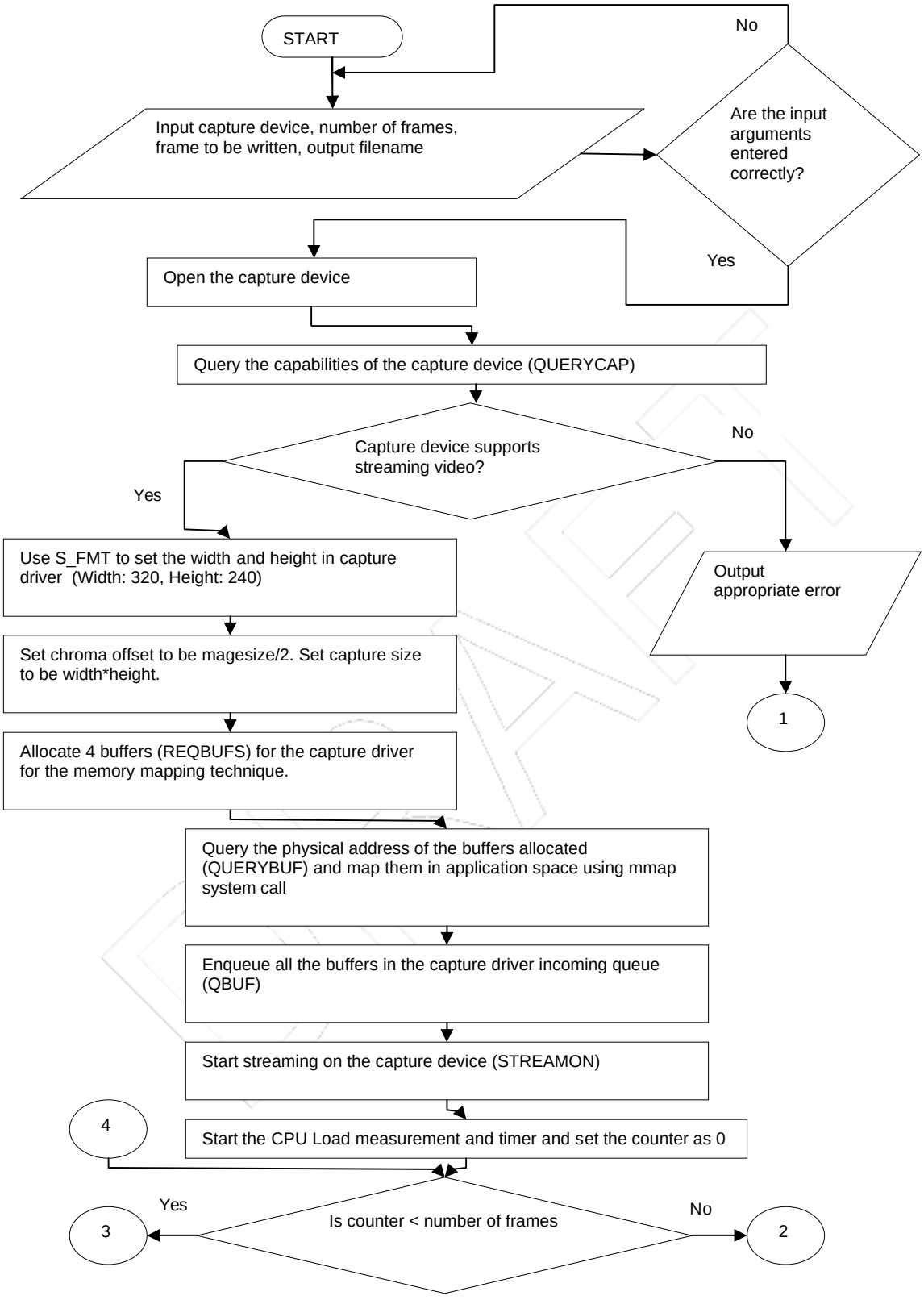
Figure 4-9. USB ISO Video Capture Flow Diagram

The USB ISO Video Capture performance is measured as number of frames captured per second for the uvcdriver.

The pspTest tool allows you to configure the following:

1. Capture Device
2. Number of frames to be captured
3. Number of the frames to be written
4. Output YUV file name where the frame is written

Figure 4-10 shows the flow chart for the function `usbisovideo_perf()`.


START
Input capture device, number of frames, frame to be written, output filename
Are the input arguments entered correctly?
No
Yes
Open the capture device
Query the capabilities of the capture device (QUERYCAP)
Capture device supports streaming video?
No
Yes
Use S_FMT to set the width and height in capture driver (Width: 320, Height: 240)
Output appropriate error
1
Set chroma offset to be magesize/2. Set capture size to be width*height.
Allocate 4 buffers (REQBUFS) for the capture driver for the memory mapping technique.
Query the physical address of the buffers allocated (QUERYBUF) and map them in application space using mmap system call
Enqueue all the buffers in the capture driver incoming queue (QBUF)
Start streaming on the capture device (STREAMON)
4
Start the CPU Load measurement and timer and set the counter as 0
Yes
Is counter < number of frames
No
3
2

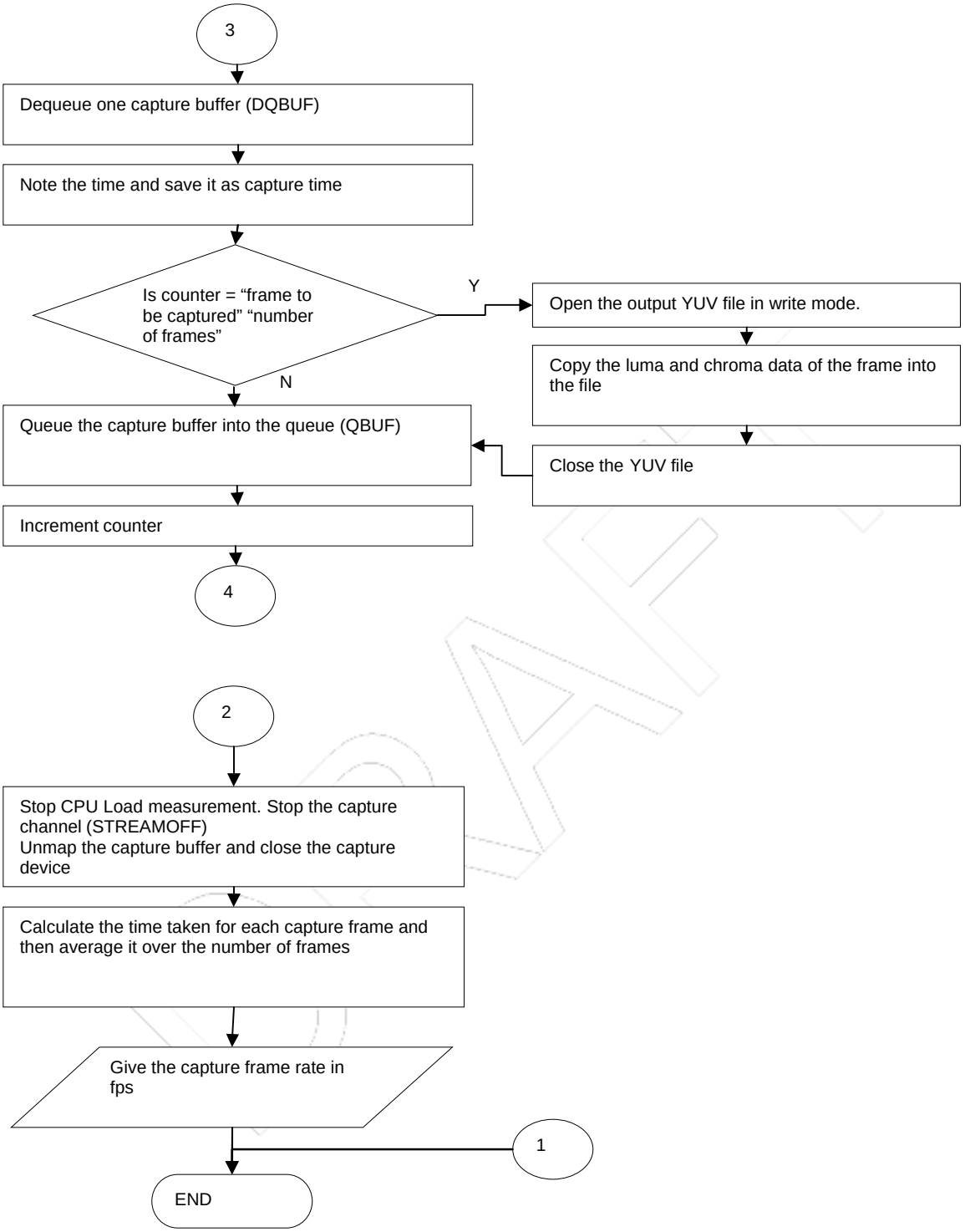


Figure 4-10. Flow chart of USB ISO Video Performance

4.4 Video Performance Tests

4.4.1 Capture Performance Tests

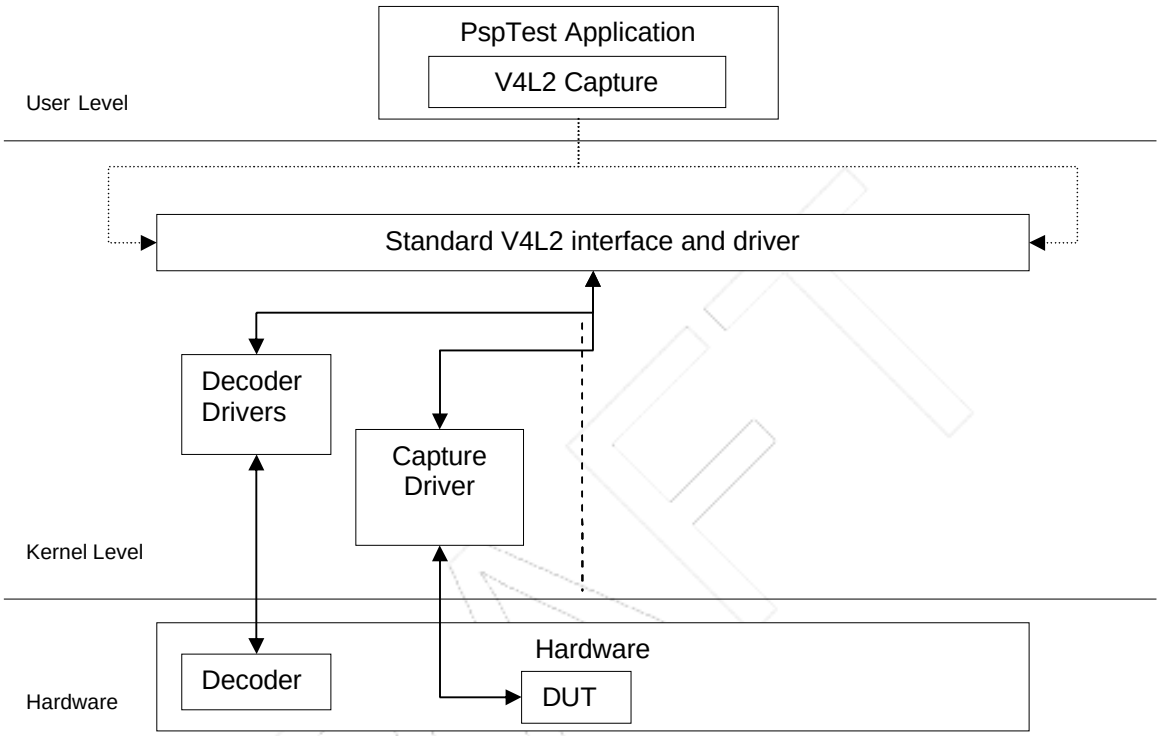


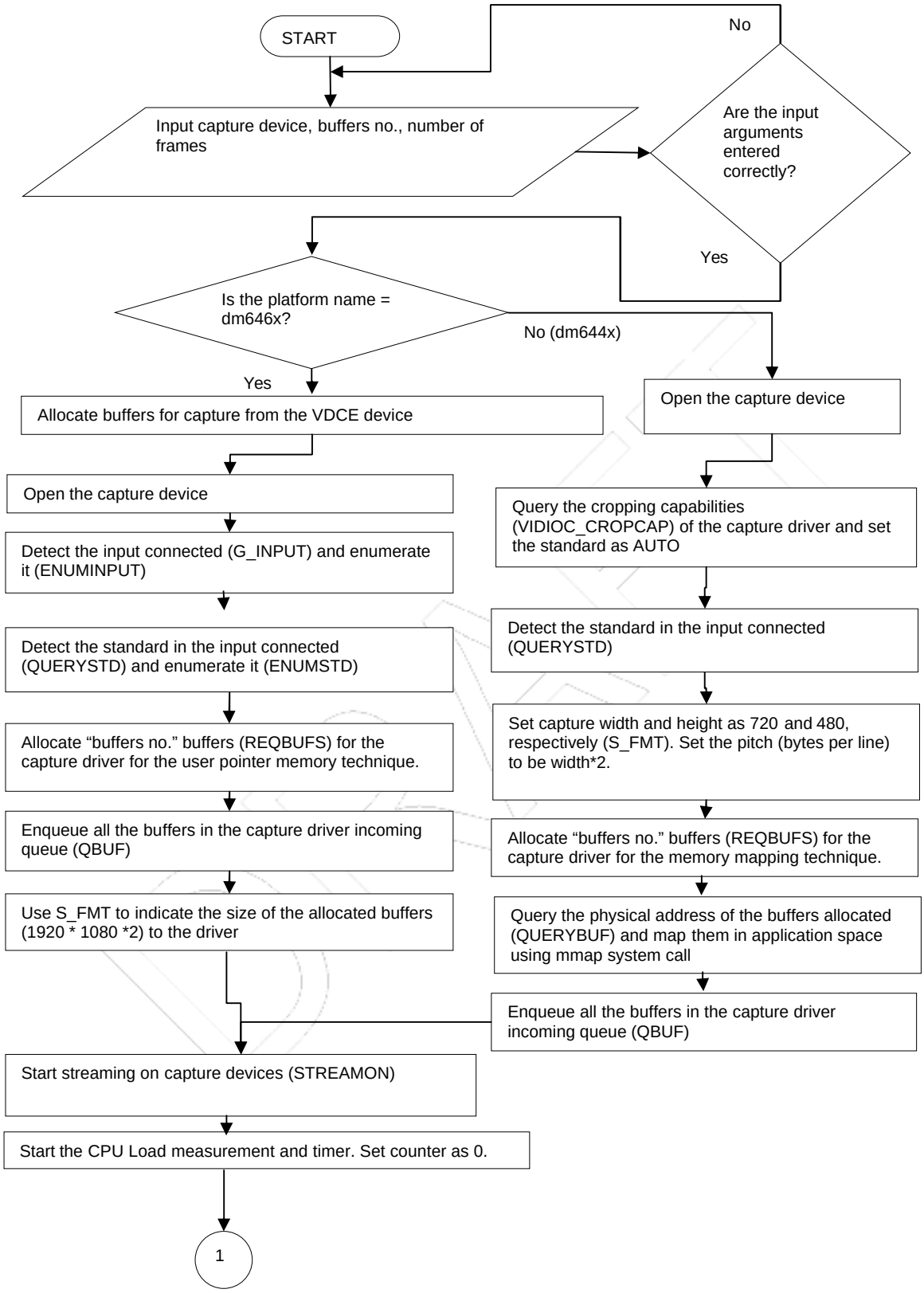
Figure 4-11. Video Capture IO Flow Diagram

The Video Capture performance for Video drivers like VPIF and VPSS is measured as number of frames captured per second.

The pspTest tool allows you to configure the following:

1. Capture device
2. Number of buffers
3. Number of frames captured

Figure 4-12 shows the flow chart for the function `v4l2capture_perf()`.


START
Input capture device, buffers no., number of frames
Are the input arguments entered correctly?
No
Yes
Is the platform name = dm646x?
No (dm644x)
Yes
Allocate buffers for capture from the VDCE device
Open the capture device
Detect the input connected (G_INPUT) and enumerate it (ENUMINPUT)
Detect the standard in the input connected (QUERYSTD) and enumerate it (ENUMSTD)
Allocate “buffers no.” buffers (REQBUFS) for the capture driver for the user pointer memory technique.
Enqueue all the buffers in the capture driver incoming queue (QBUF)
Use S_FMT to indicate the size of the allocated buffers (1920 * 1080 *2) to the driver
Open the capture device
Query the cropping capabilities (VIDIOC_CROPCAP) of the capture driver and set the standard as AUTO
Detect the standard in the input connected (QUERYSTD)
Set capture width and height as 720 and 480, respectively (S_FMT). Set the pitch (bytes per line) to be width*2.
Allocate “buffers no.” buffers (REQBUFS) for the capture driver for the memory mapping technique.
Query the physical address of the buffers allocated (QUERYBUF) and map them in application space using mmap system call
Enqueue all the buffers in the capture driver incoming queue (QBUF)
Start streaming on capture devices (STREAMON)
Start the CPU Load measurement and timer. Set counter as 0.
1

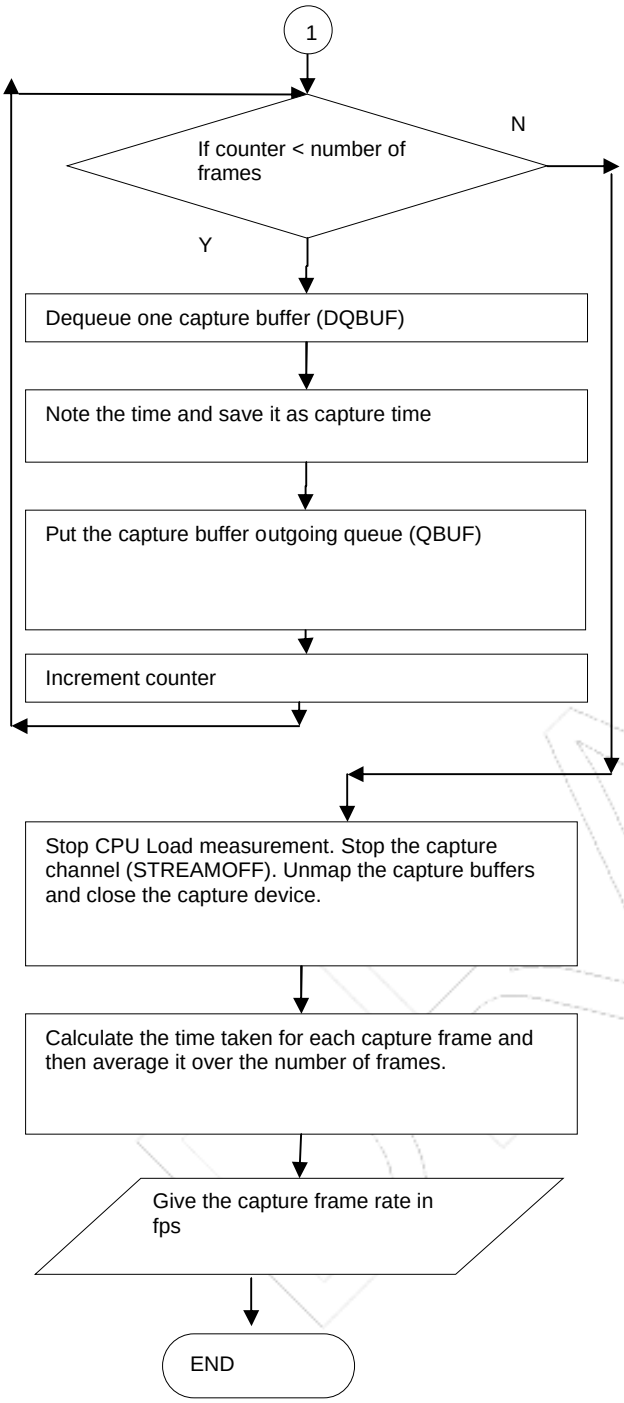


Figure 4-12. Flow Chart of Video Capture Performance

4.4.2 Display Performance Tests

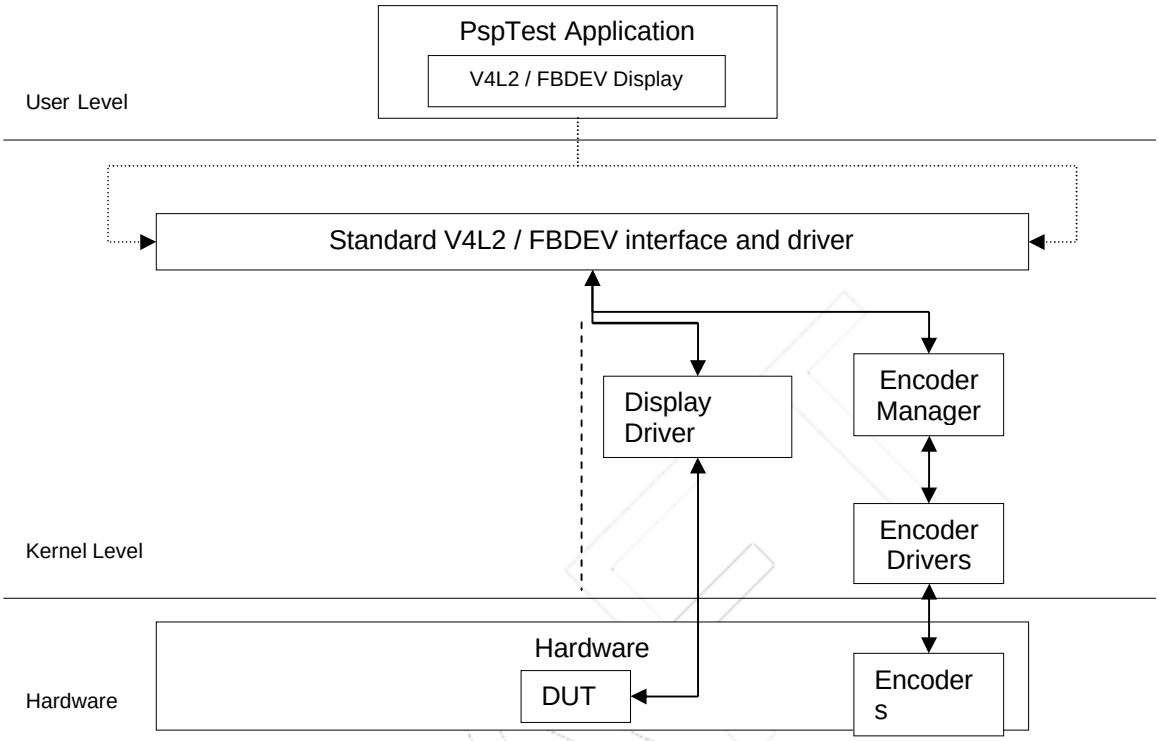


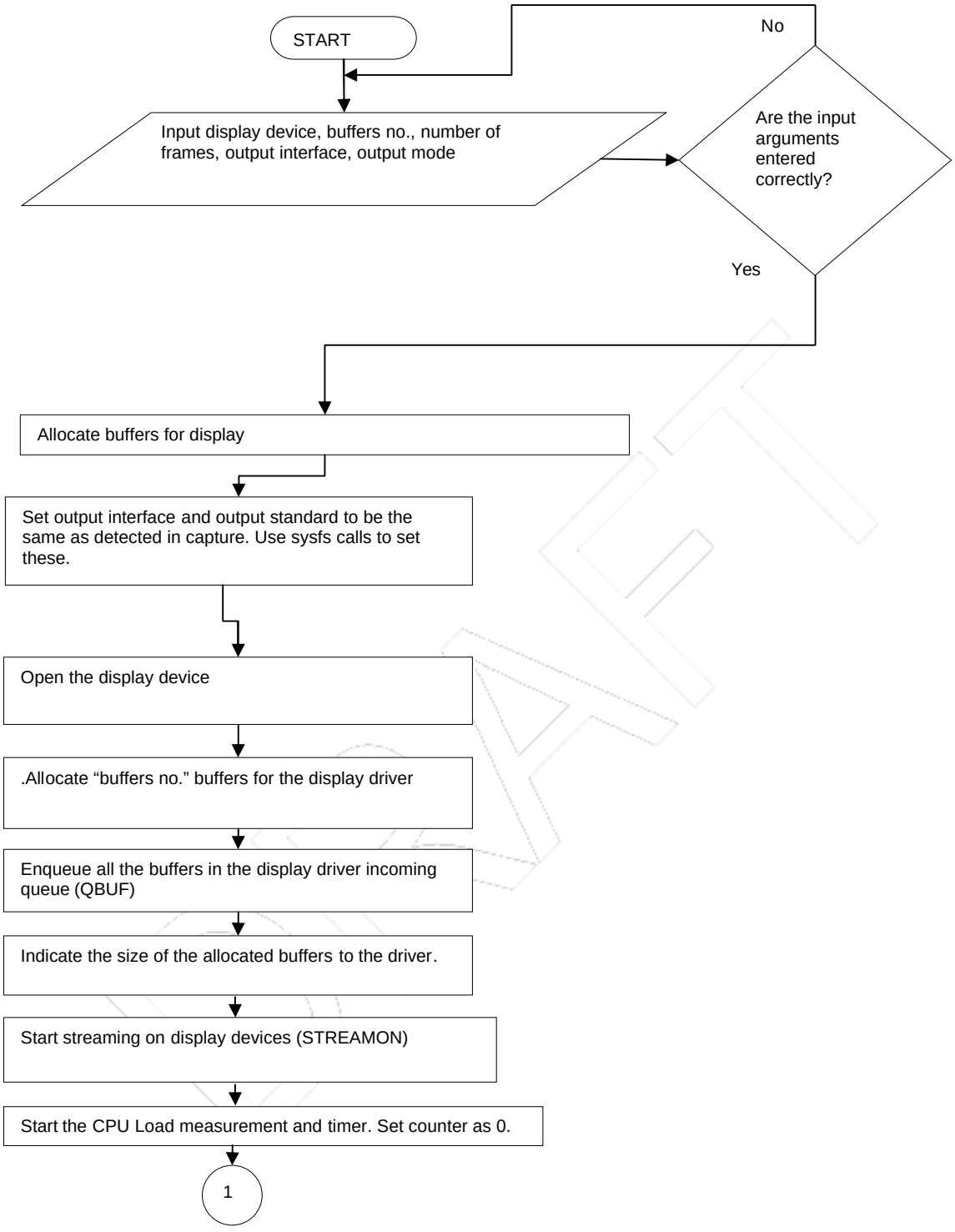
Figure 4-133. Video Display IO Flow Diagram

The Video Display performance for Video drivers like VPIF and VPSS is measured as number of frames displayed per second.

The pspTest tool allows you to configure the following:

1. Display device
2. Number of buffers
3. Number of frames captured and displayed
4. Output Interface
5. Output mode / resolution

Figure 4-12 shows the flow chart for the function `display_perf()`.


START
Input display device, buffers no., number of frames, output interface, output mode
Are the input arguments entered correctly?
No
Yes
Allocate buffers for display
Set output interface and output standard to be the same as detected in capture. Use sysfs calls to set these.
Open the display device
.Allocate “buffers no.” buffers for the display driver
Enqueue all the buffers in the display driver incoming queue (QBUF)
Indicate the size of the allocated buffers to the driver.
Start streaming on display devices (STREAMON)
Start the CPU Load measurement and timer. Set counter as 0.
1

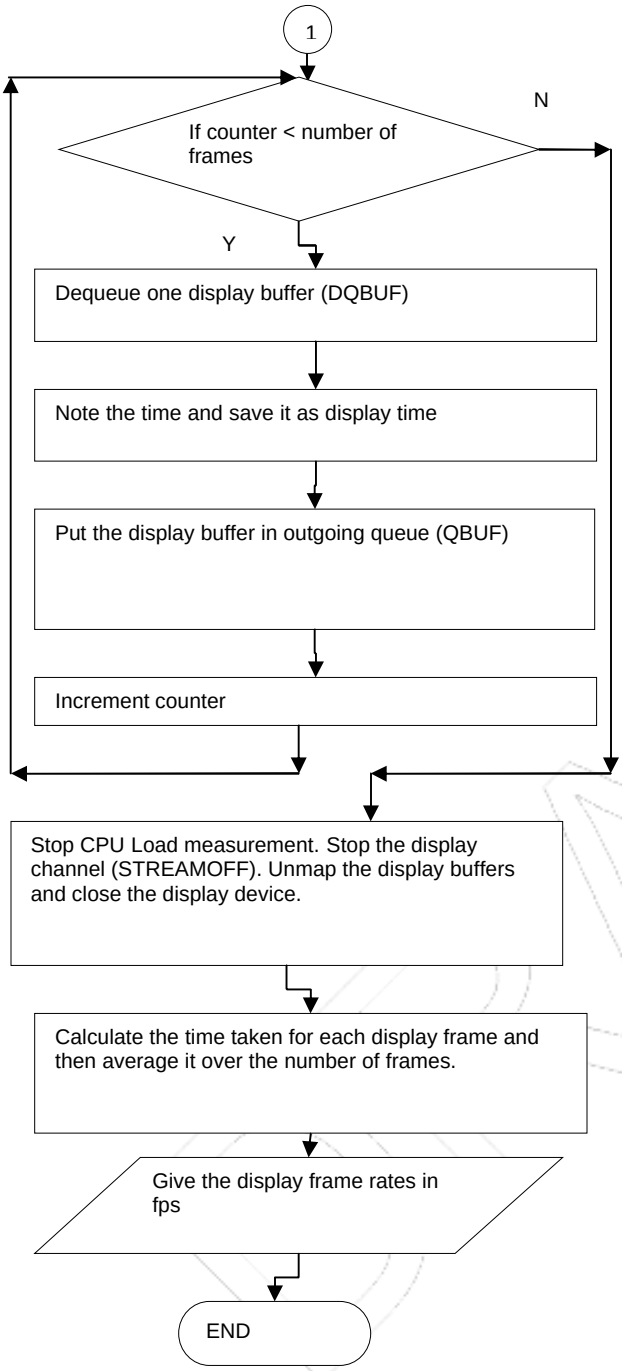


Figure 4-144. Flow Chart of Video Display Performance

4.5 VDCE Performance Tests

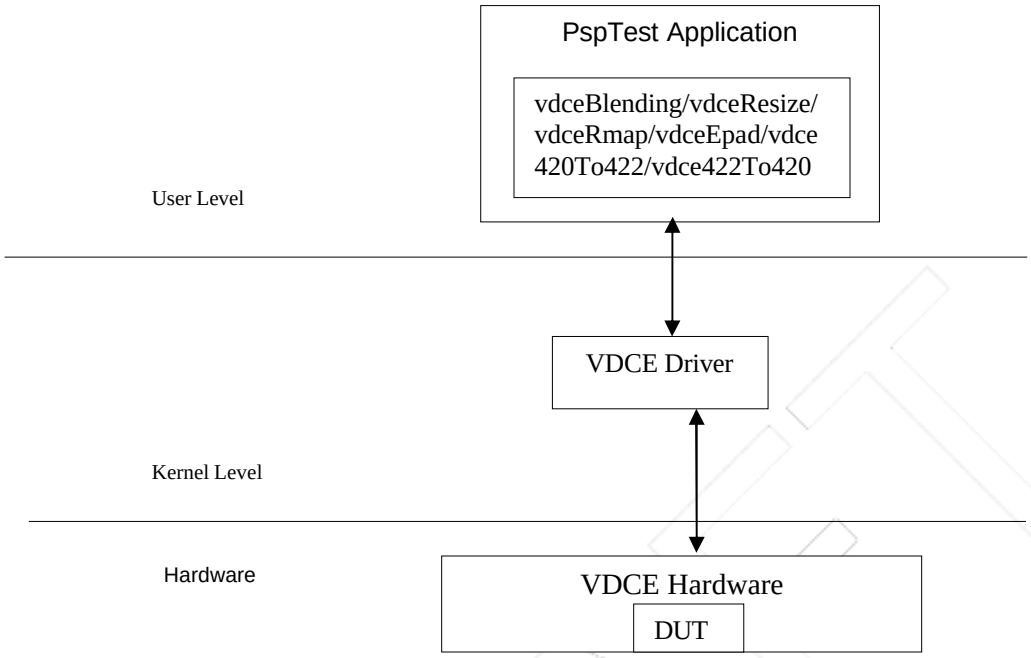


Figure 4-15. VDCE IO Flow Diagram

4.5.1 Resize Performance

The resize performance for VDCE driver is measured as time taken to perform resize operation on a given input frame.

You need to provide the following inputs to the pspTest tool to calculate time taken for resizing operation:

1. Device node

   For example: /dev/DavinciHD_vdce

2. Input image width

   For example: 1920

3. Input image height

   For example: 1080

4. Input image name

   For example: 1080i.yuv

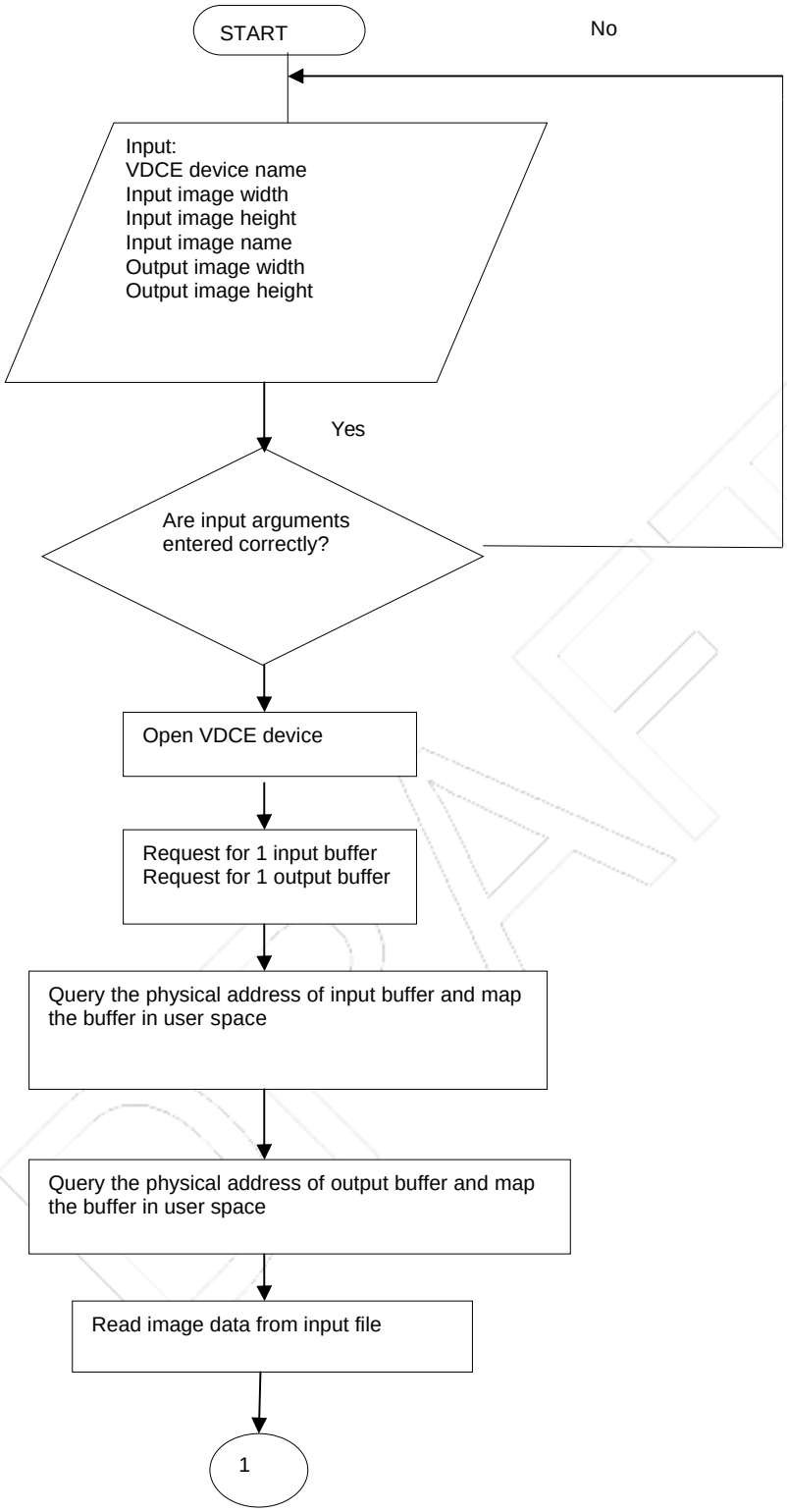
5. Output image width

   For example: 720

6. Output image height

   For example: 480

Figure 4-16 shows the flow chart for the function `vdce_resize ()`.


START
No
Input:
VDCE device name
Input image width
Input image height
Input image name
Output image width
Output image height
Yes
Are input arguments entered correctly?
Open VDCE device
Request for 1 input buffer
Request for 1 output buffer
Query the physical address of input buffer and map the buffer in user space
Query the physical address of output buffer and map the buffer in user space
Read image data from input file
1

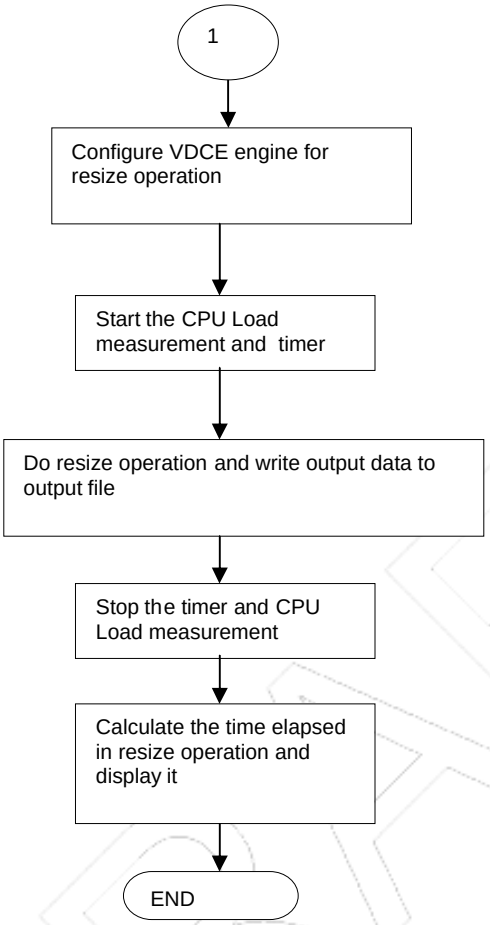


Figure 4-16. Flow Chart of VDCE Resize

4.5.2 Blending Performance

The blending performance for VDCE driver is measured as time taken to perform blending operation on a given input frame. You need to provide the following inputs to the pspTest tool to calculate time taken for blending operation:

1. Device node

   For example, /dev/DavinciHD_vdce.

2. Input image width

   For example, 1920.

3. Input image height

   For example, 1080.

4. Input image name

   For example, 1080i.yuv.

5. Bitmap width

   For example, 256.

6. Bitmap height

   For example, 160.

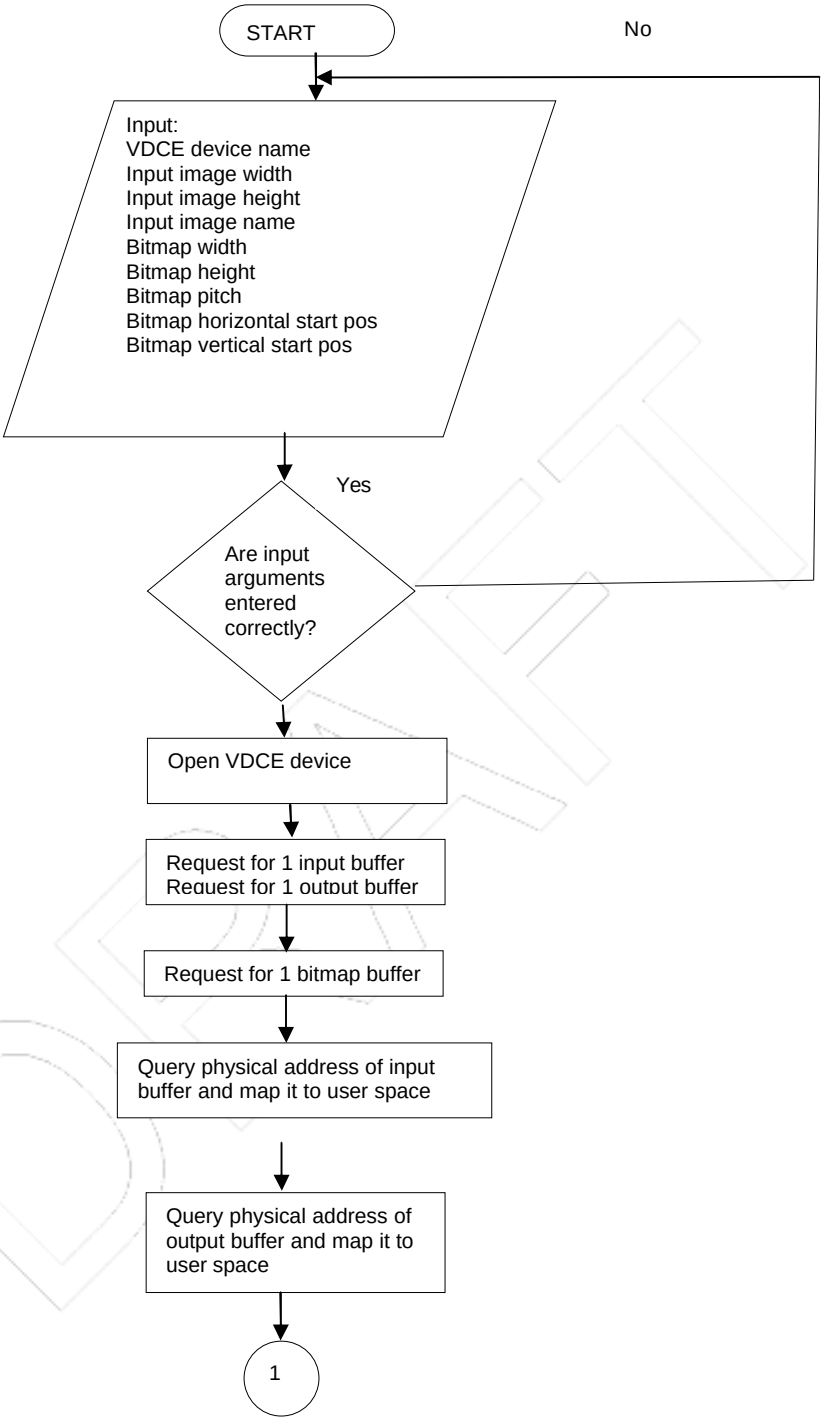
7. Bitmap pitch

   For example, 64.

8. Bitmap horizontal start position

9. Bitmap vertical start position

Figure 4-17 shows the flow chart for the function `vdce_blending ()`.


START
No
Input:
VDCE device name
Input image width
Input image height
Input image name
Bitmap width
Bitmap height
Bitmap pitch
Bitmap horizontal start pos
Bitmap vertical start pos
Yes
Are input arguments entered correctly?
Open VDCE device
Request for 1 input buffer
Request for 1 output buffer
Request for 1 bitmap buffer
Query physical address of input buffer and map it to user space
Query physical address of output buffer and map it to user space
1

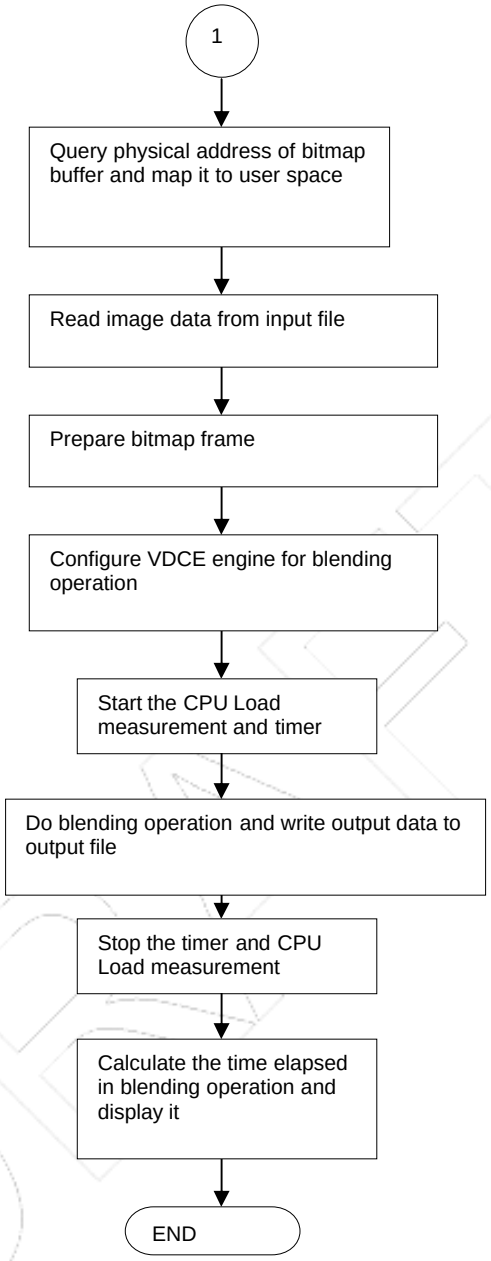


Figure 4-17. Flow Chart of VDCE Blending

4.5.3 Edge Padding Performance

The edge padding performance for VDCE driver is measured as time taken to perform edge padding operation on a given input frame. You need to provide the following inputs to the pspTest tool to calculate time taken for edge padding operation:

1. Device node

   For example, /dev/DavinciHD_vdce.

2. Input image width

For example, 1920.

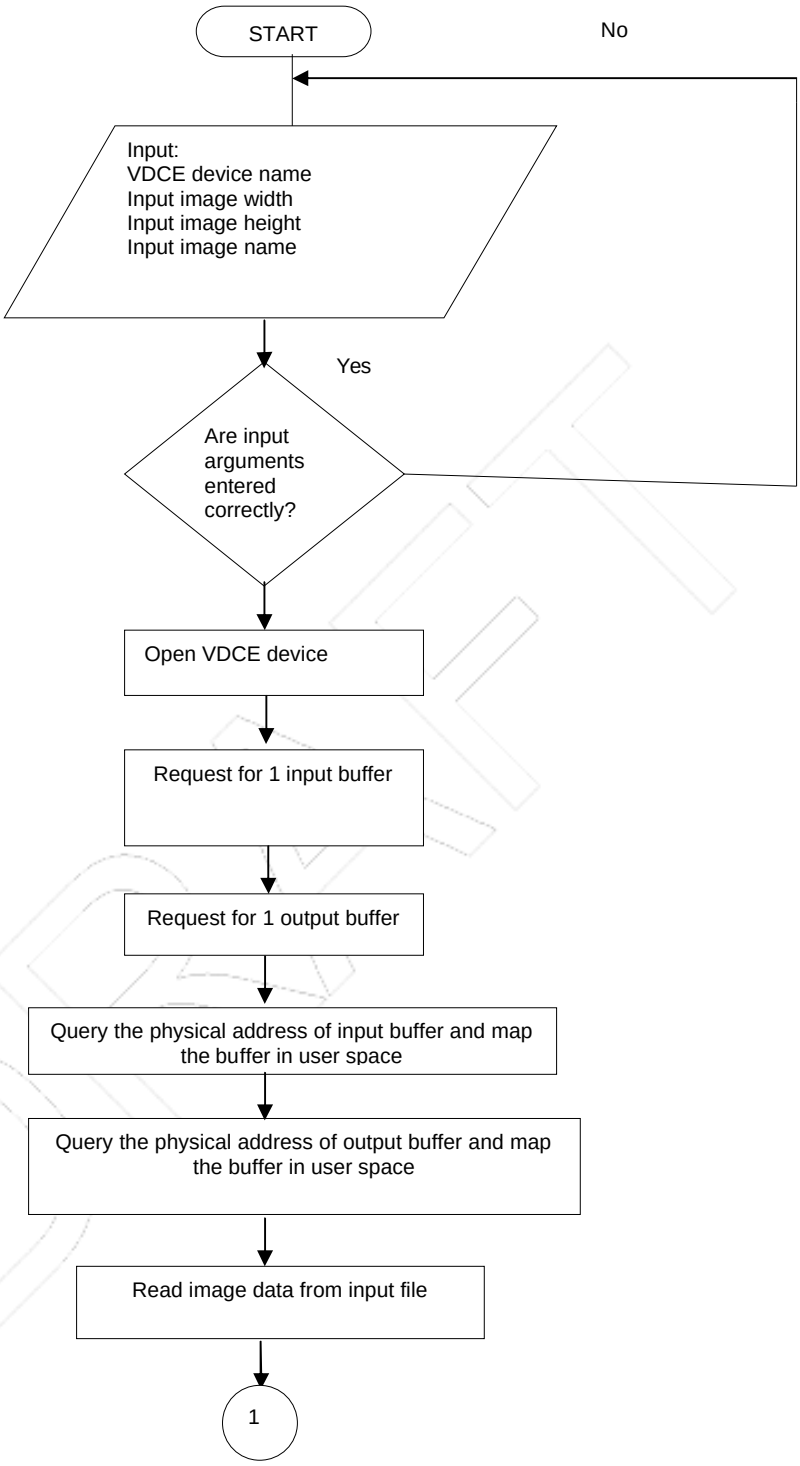
3. Input image height

   For example, 1080.

4. Input image name

   For example, 1080i.yuv.

Figure 4-18 shows the flow chart for the function `vdce_epad()`.


START
No
Input:
VDCE device name
Input image width
Input image height
Input image name
Yes
Are input arguments entered correctly?
Open VDCE device
Request for 1 input buffer
Request for 1 output buffer
Query the physical address of input buffer and map the buffer in user space
Query the physical address of output buffer and map the buffer in user space
Read image data from input file
1

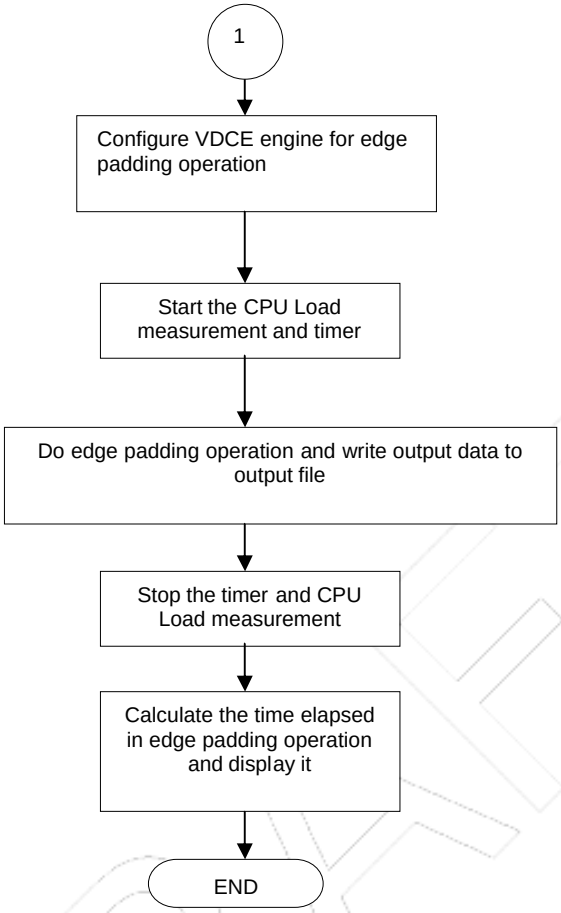


Figure 4-18. Flow Chart of VDCE Edge Padding

4.5.4 Range Mapping Performance

The range mapping performance for VDCE driver is measured as time taken to perform range mapping operation on a given input frame. You need to provide the following inputs to the pspTest tool to calculate time taken in range mapping operation:

1. Device node

   For example, /dev/DavinciHD_vdce.

2. Input image width

   For example, 1920.

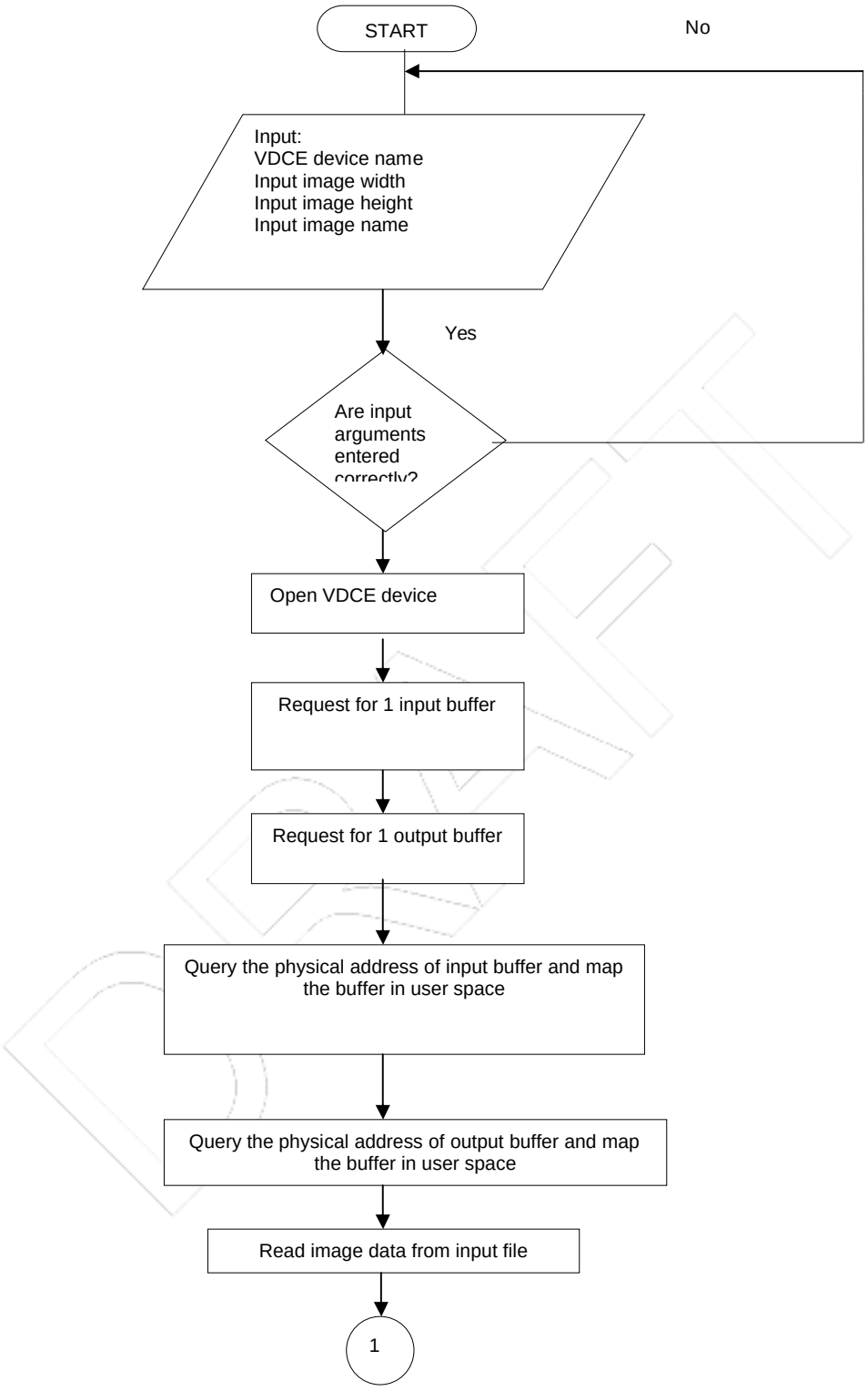
3. Input image height

   For example, 1080.

4. Input image name

   For example, 1080i.yuv.

Figure 4-19 shows the flow chart for the function `vdce_rmap ()`.


START
No
Input:
VDCE device name
Input image width
Input image height
Input image name
Yes
Are input arguments entered correctly?
Open VDCE device
Request for 1 input buffer
Request for 1 output buffer
Query the physical address of input buffer and map the buffer in user space
Query the physical address of output buffer and map the buffer in user space
Read image data from input file
1

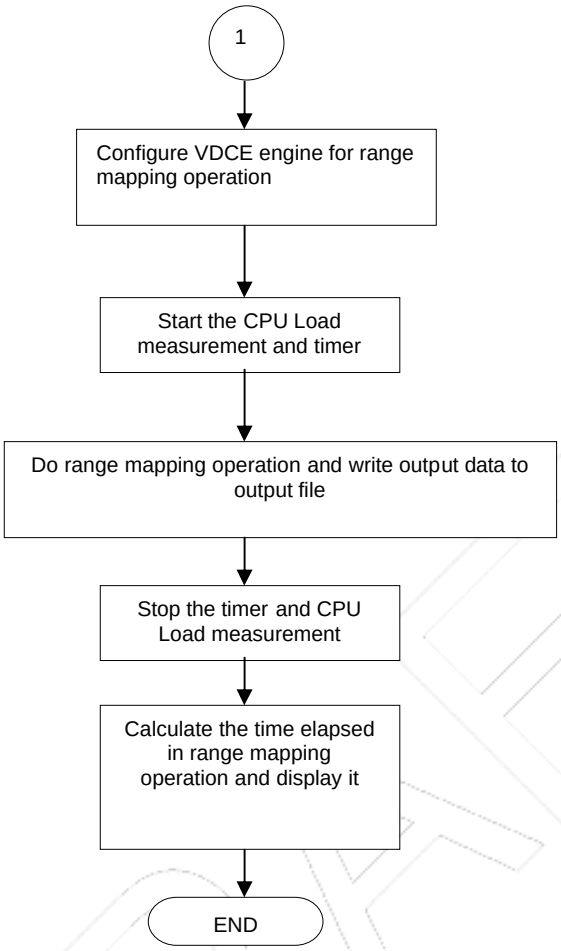


Figure 4-19.Flow Chart of VDCE Range mapping

4.5.5 Chrominance Conversion (YUV 420 to YUV 422) Performance

The CCV (420 to 422) performance for VDCE driver is measured as time taken to perform CCV operation on a given YUV 420 image. You need to provide the following inputs to the pspTest tool to calculate time taken for CCV operation:

1. Device node

   For example, /dev/DavinciHD_vdce.

2. Input image width

   For example, 1920.

3. Input image height

   For example, 1080.

4. Input image name

   For example, 1080i_420.yuv.

Figure 4-20 shows the flow chart for the function `vdce_ccv420_422 ()`.

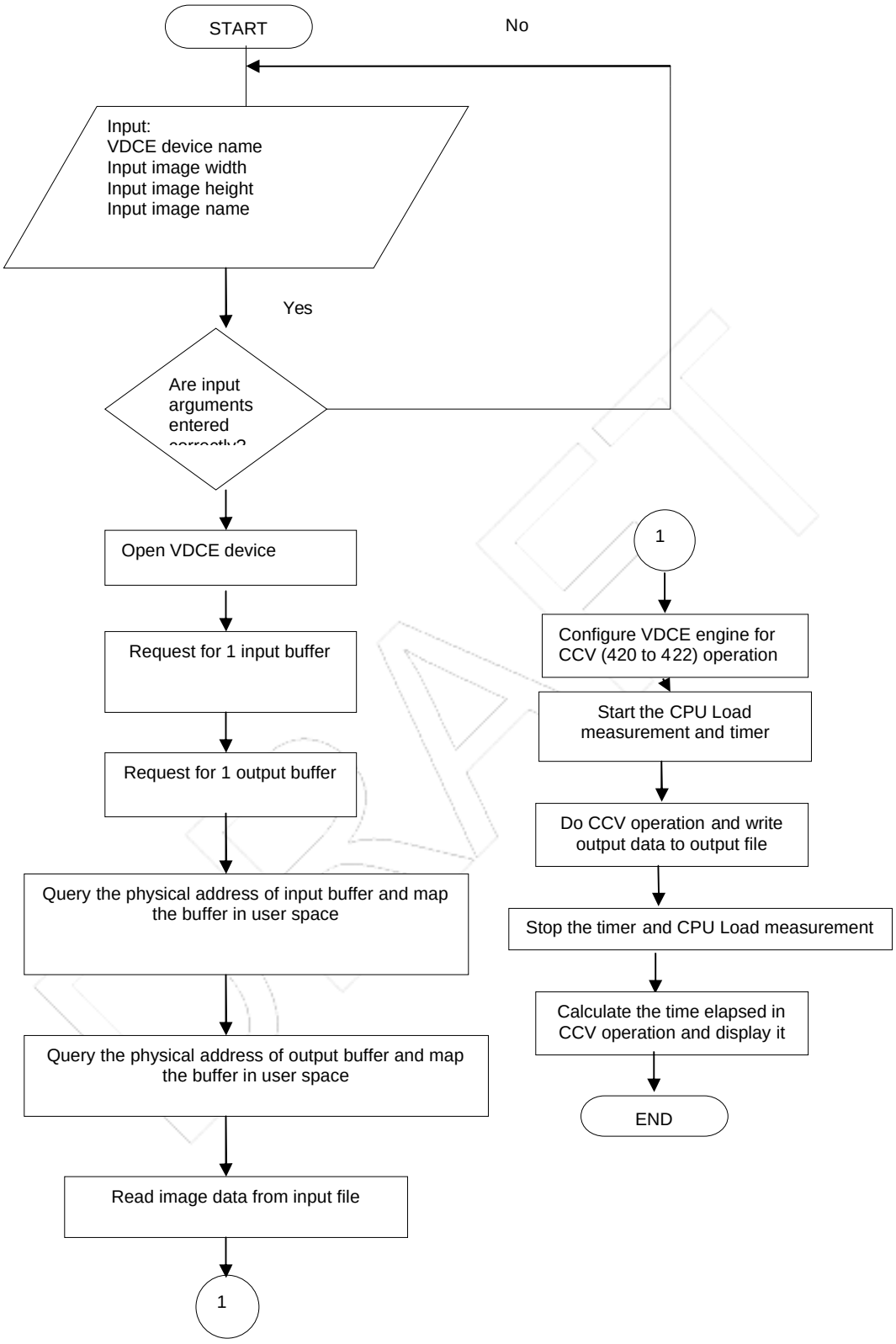


Figure 4-20. Flow chart of VDCE Chrominance Conversion (YUV 420 to YUV 422)

4.5.6 Chrominance Conversion (YUV 422 to YUV 420) Performance

The CCV (422 to 420) performance for VDCE driver is measured as time taken to perform CCV operation on a given YUV 422 image. You need to provide the following inputs to the pspTest tool to calculate time taken for CCV operation:

1. Device node

   For example, /dev/DavinciHD_vdce.

2. Input image width

   For example, 1920.

3. Input image height

   For example, 1080.

4. Input image name

   For example, 1080i_422.yuv.

Figure 4-21 shows the flow chart for the function `vdce_ccv422_420 ()`.

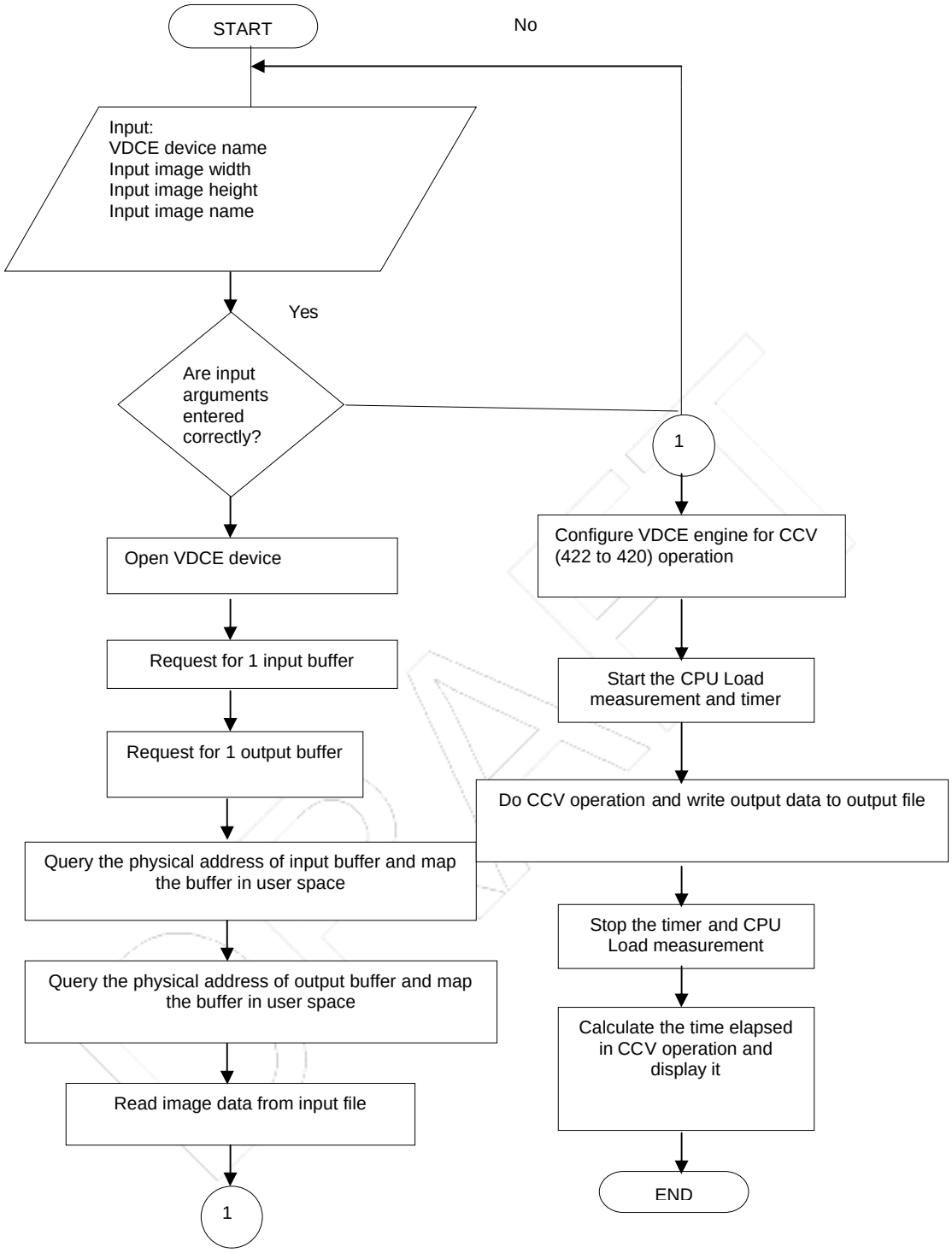


Figure 4-21. Flow Chart of VDCE Chrominance Conversion (YUV 422 to YUV 420)

4.6 I2C Performance Tests

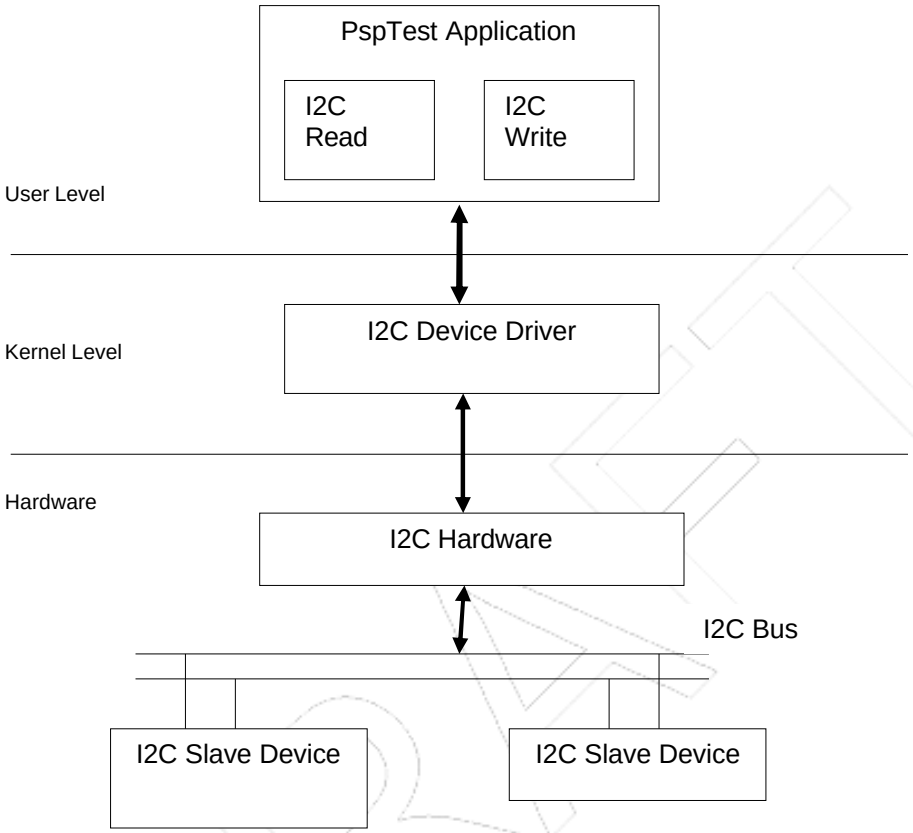


Figure 4-22. I2C Data Flow Diagram

4.6.1 I2C Read Performance:

The read performance for I2C is measured as time taken in micro seconds to read from I2C Slave (EEPROM) for a given buffer size.

The pspTest allows you to configure the buffer size and application buffer size for read operation.

Note:

The total buffer size should be greater than application buffer size.

Figure 4-23 shows the flow chart for the function `throughputI2cRead ()`.

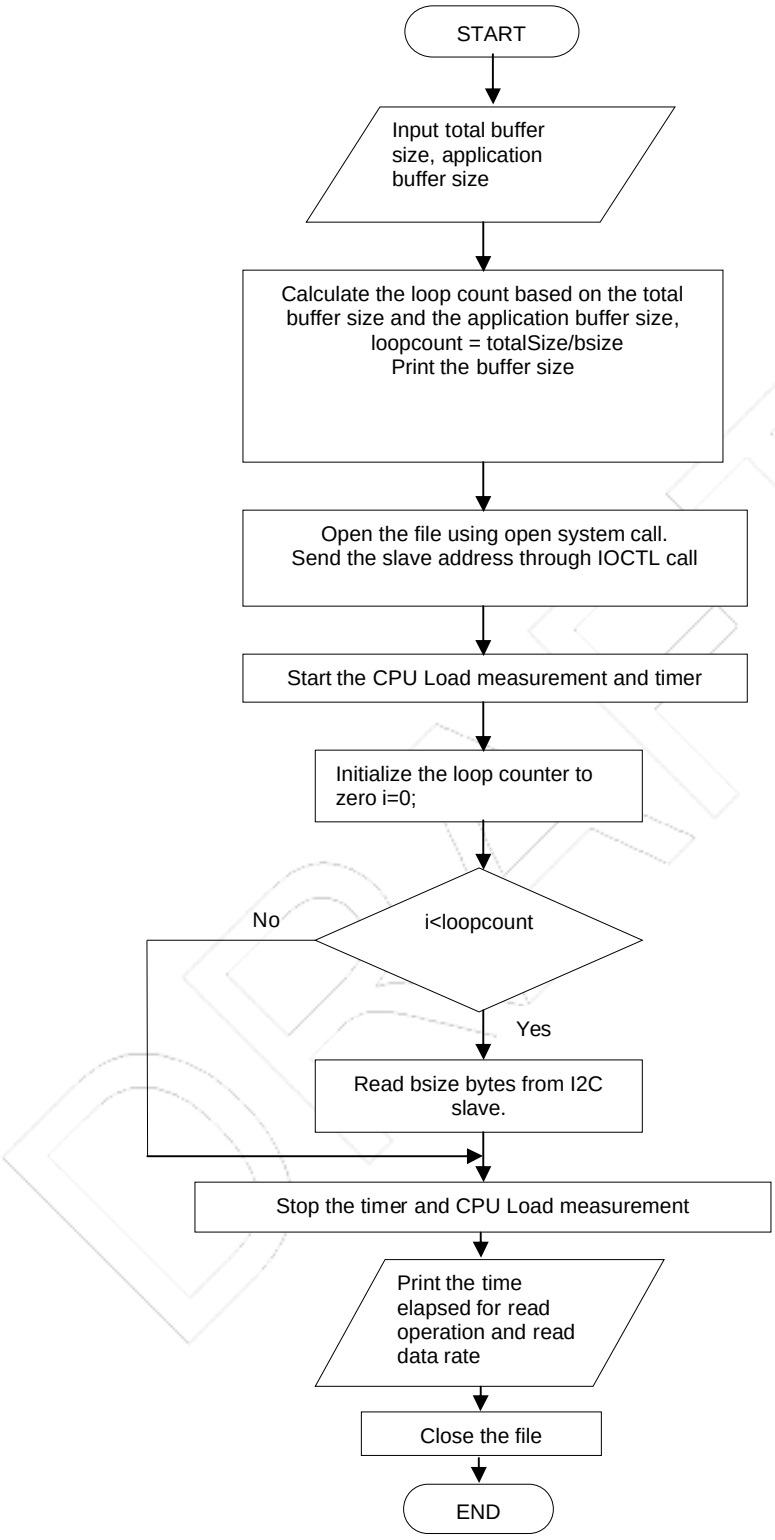


Figure 4-23. Flow Chart of I2C Read

4.6.2 I2C Write Performance:

The write performance for I2C is measured as time taken in micro seconds to write to I2C Slave (EEPROM) for a given total buffer size.

The pspTest allows you to configure the buffer size and application buffer size for write operation.

Note:

The total buffer size should be greater than application buffer size.

Figure 4-24 shows the flow chart for the function `throughputI2cWrite()`.

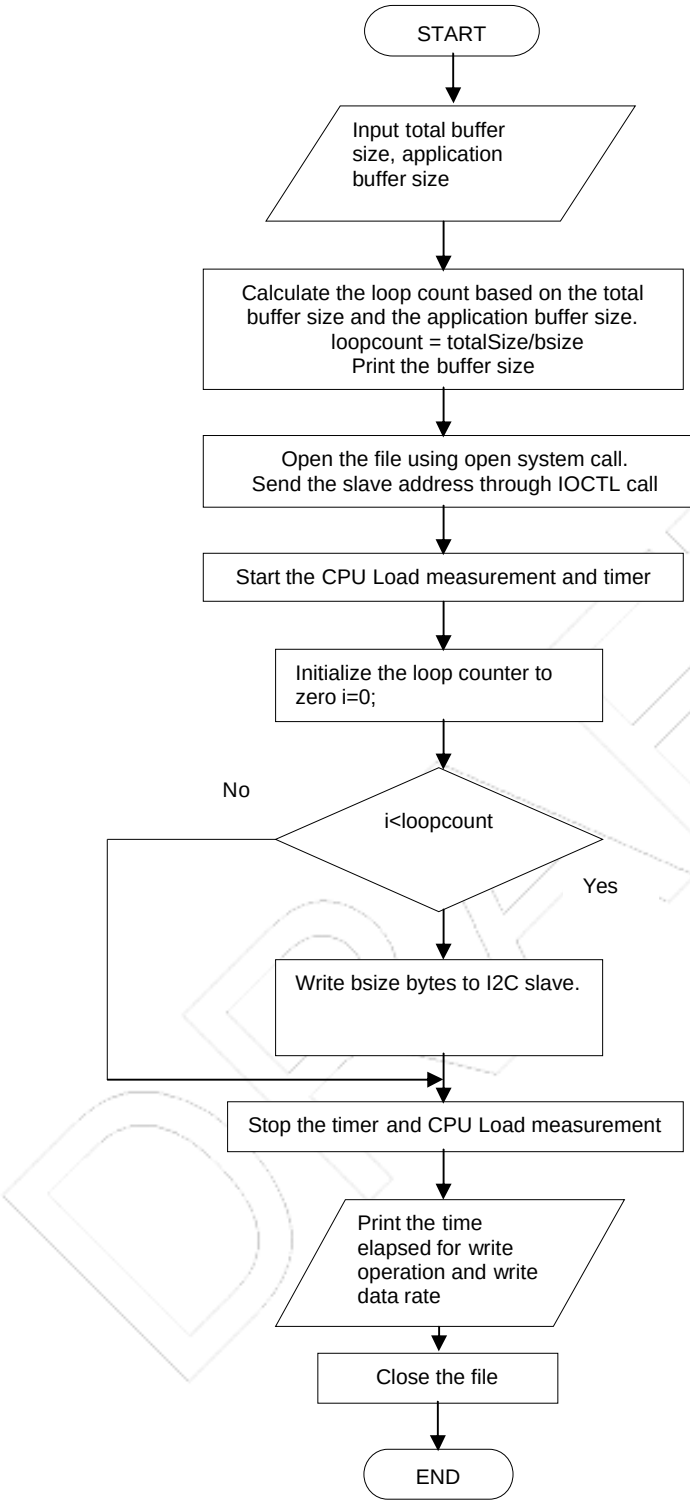


Figure 4-24. Flow chart of I2C Write

4.7 SPI Performance Tests

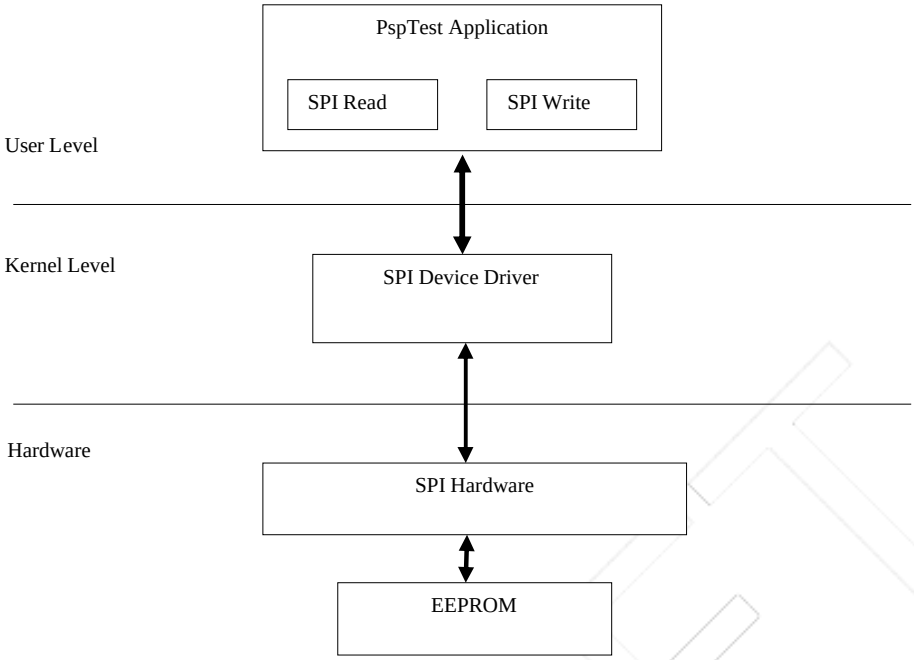


Figure 4-25. SPI Data Flow Diagram

4.7.1 SPI Read Performance:

The read performance for SPI is measured as time taken in micro seconds to read from SPI Slave (EEPROM) for a given buffer size.

The pspTest allows you to configure the buffer size used to read a file.

Figure 4-26 shows the flow chart for the function `throughputSpiRead ()`.

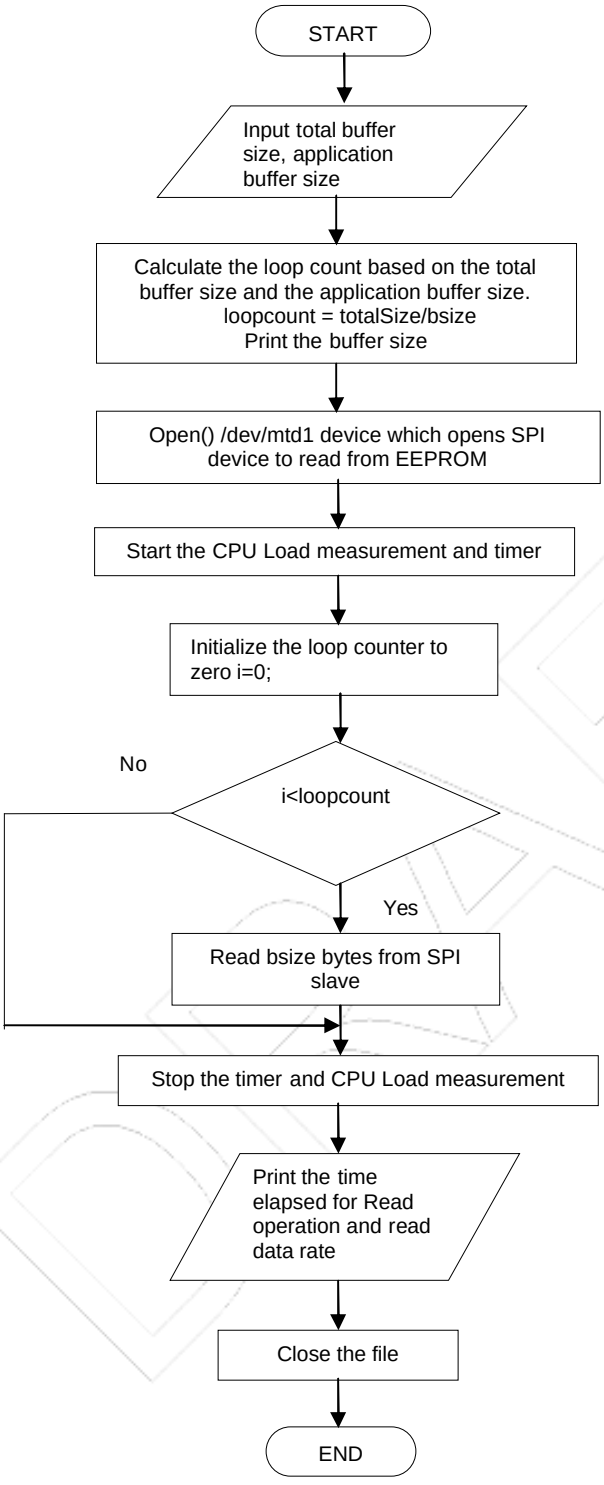


Figure 4-26. Flow Chart of SPI Read

4.7.2 SPI Write Performance

The write performance for SPI is measured as time taken in micro seconds to write to SPI Slave (EEPROM) for a given total buffer size.

The pspTest allows you to configure the total buffer size used to read a file.

Figure 4-27 shows the flow chart for the function `throughputSPIWrite()`.

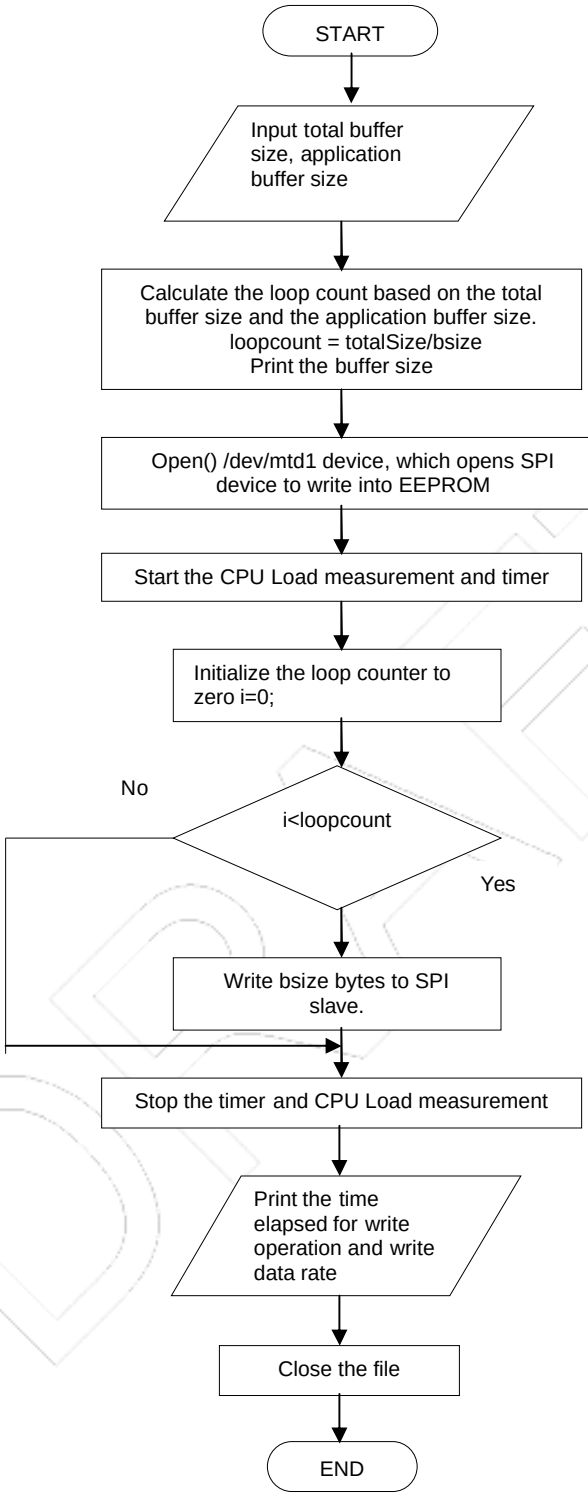


Figure 4-27. Flow Chart of SPI Write

4.8 EDMA/QDMA Performance Tests

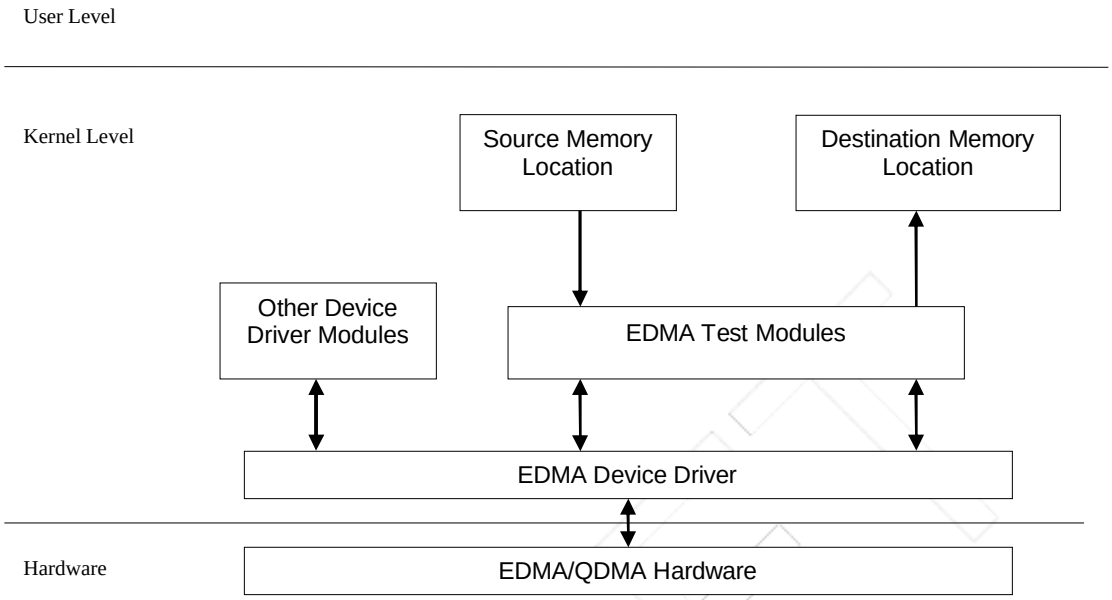


Figure 4-28. EDMA/QDMA Data Flow Diagram

4.8.1 EDMA A Sync Incremental Mode Data Transfer

The performance measurement of data transfer using EDMA channels is done by transferring 64Kbytes of data (65535 Bytes) using DMA channels and by measuring the time taken to perform the transfer. Table 4-1 shows the different combinations of A, B, and C counts used to achieve 64K bytes of transfer.

A Count	B Count	C Count	Total Size (bytes)
1024	64	1	**65535**
4096	16	1	**65535**
8192	8	1	**65535**
16384	4	1	**65535**
32767	2	1	**65535**
65535	**1**	**1**	65535

Table 4-1. A, B, and C Counts for EDMA A Sync Incremental Mode

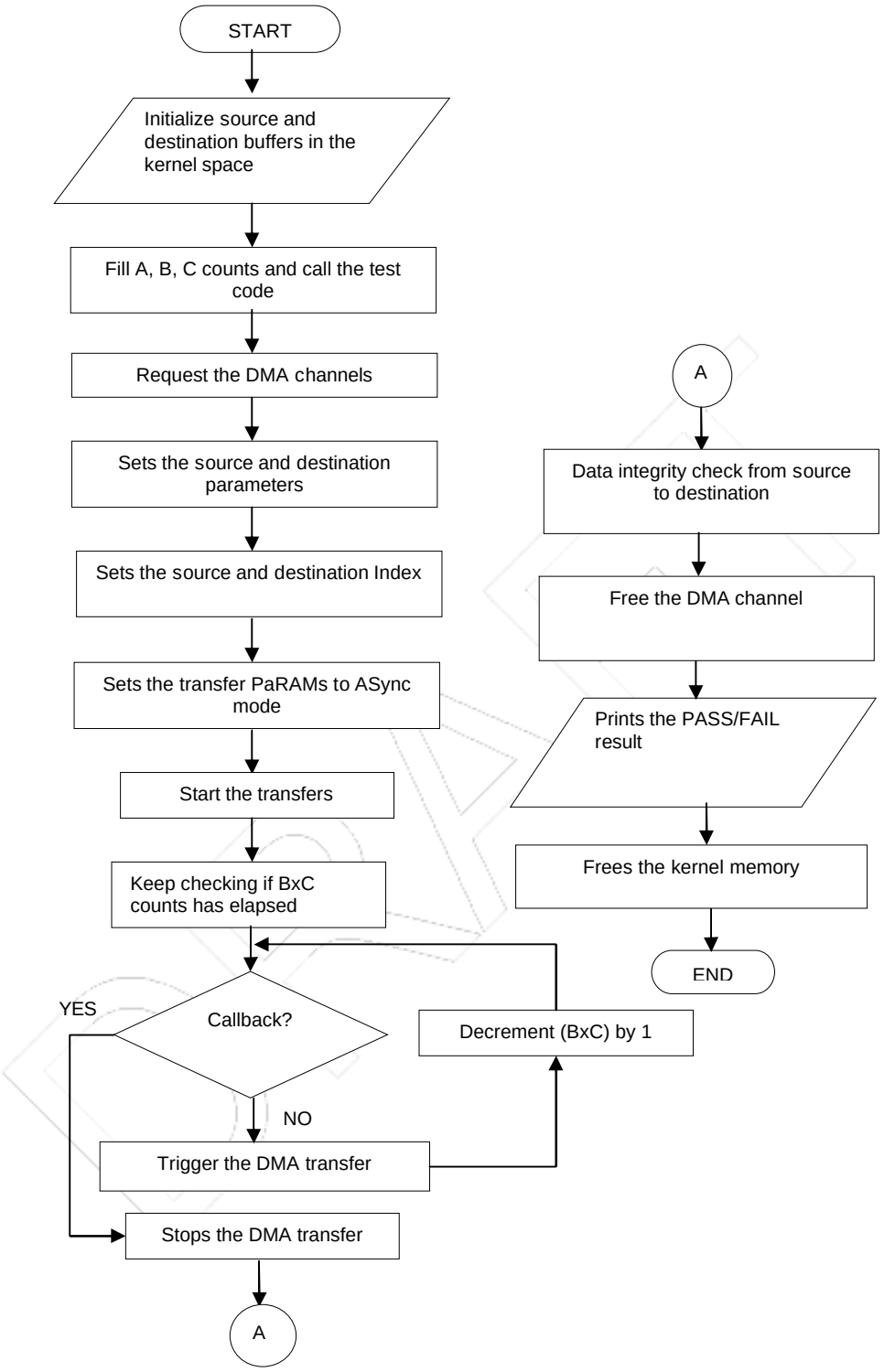


Figure 4-29. Flow Chart of EDMA A Sync Incremental Mode

4.8.2 EDMA AB Sync Incremental Mode Data Transfer

The performance measurement of data transfer using EDMA channels is done by transferring 64Kbytes of data (65535 Bytes) using DMA channels and by measuring the time taken to perform the transfer. Table 4-2 shows the different combinations of A, B, and C counts used to achieve 64K bytes of transfer.

A Count	B Count	C Count	Total Size (Bytes)
1024	64	1	**65535**
4096	16	1	**65535**
8192	8	1	**65535**
16384	4	1	**65535**
32767	2	1	**65535**
65535	**1**	**1**	**65535**

Table 4-2. A, B, and C Counts for EDMA AB Sync Incremental Mode

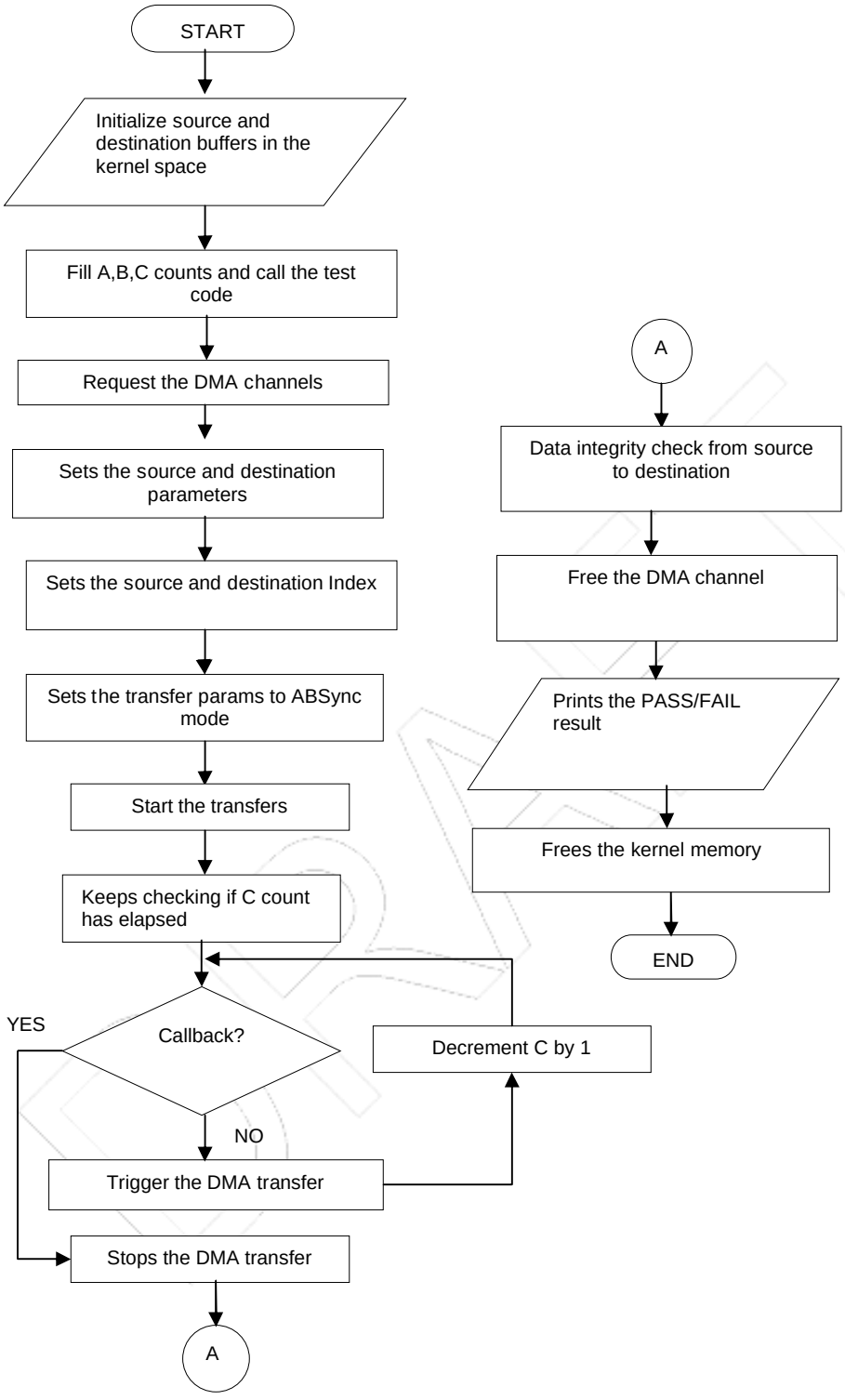


Figure 4-30. Flow Chart of EDMA AB Sync Incremental Mode

4.8.3 QDMA A Sync Incremental Mode Data Transfer

The performance measurement of data transfer using QDMA channels is done by transferring 64Kbytes of data (65535 bytes) using QDMA channels and by measuring the time taken to perform the transfer. Table 4-3 shows

the different combinations of A, B, and C counts used to achieve 64K bytes of transfer.

A Count	B Count	C Count	Total Size (Bytes)
1024	64	1	**65535**
4096	16	1	**65535**
8192	8	1	**65535**
16384	4	1	**65535**
32767	2	1	**65535**
65535	**1**	**1**	**65535**

Table 4-3. A, B, and C Counts for QDMA A Sync Incremental Mode

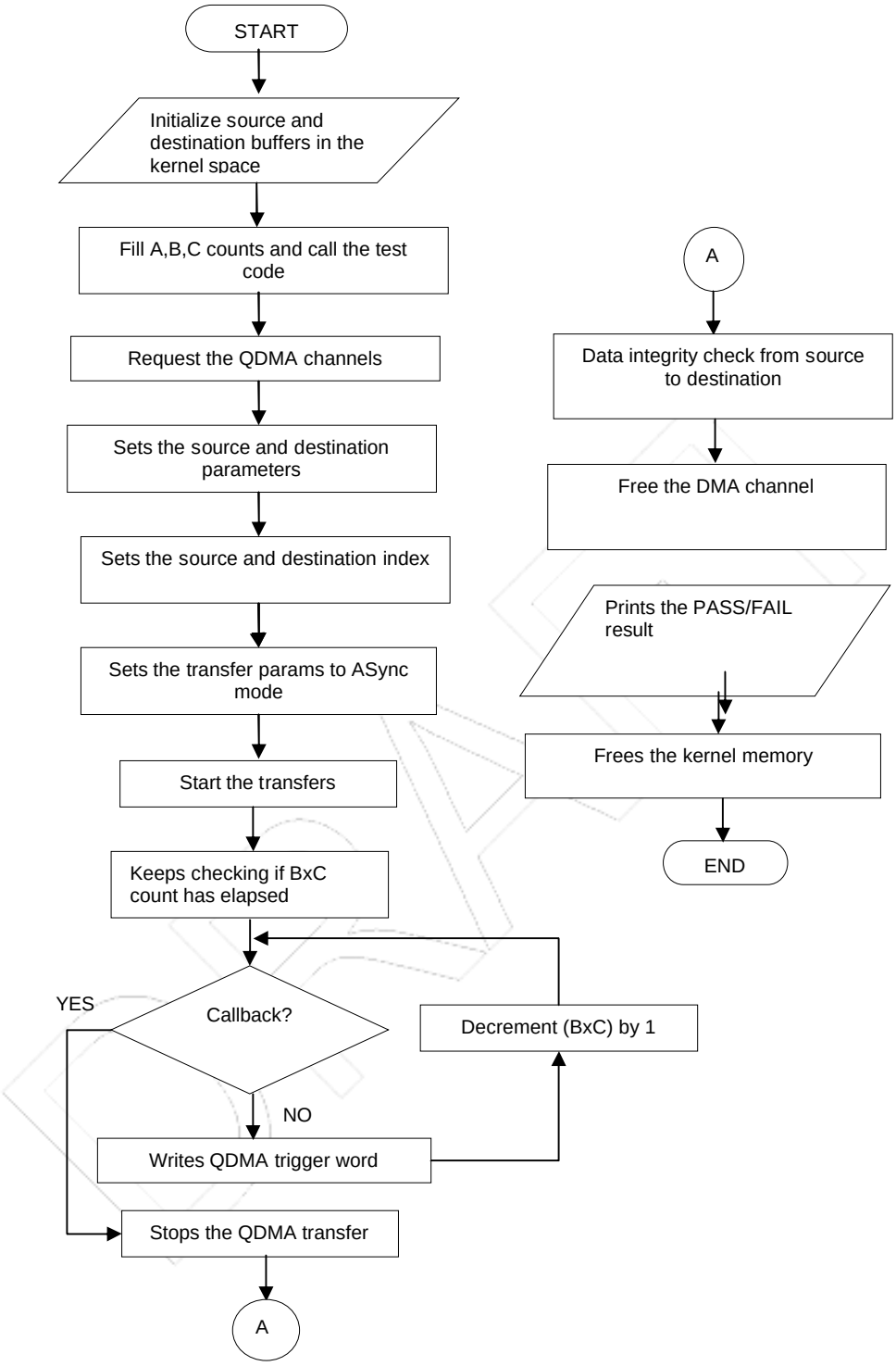


Figure 4-31. Flow Chart of QDMA A Sync Incremental Mode

4.8.4 QDMA AB Sync Incremental Mode Data Transfer

The performance measurement of data transfer using QDMA channels is done by transferring 64Kbytes of data (65535 bytes) using QDMA channels and by measuring the time taken to perform the transfer. Table 4-4 shows the different combinations of A, B, and C counts used to achieve 64K bytes of transfer.

A Count	B Count	C Count	Total Size (Bytes)
1024	64	1	**65535**
4096	16	1	**65535**
8192	8	1	**65535**
16384	4	1	**65535**
32767	2	1	**65535**
65535	**1**	**1**	**65535**

Table 4-4. A, B, and C Counts for QDMA AB Sync Incremental Mode.

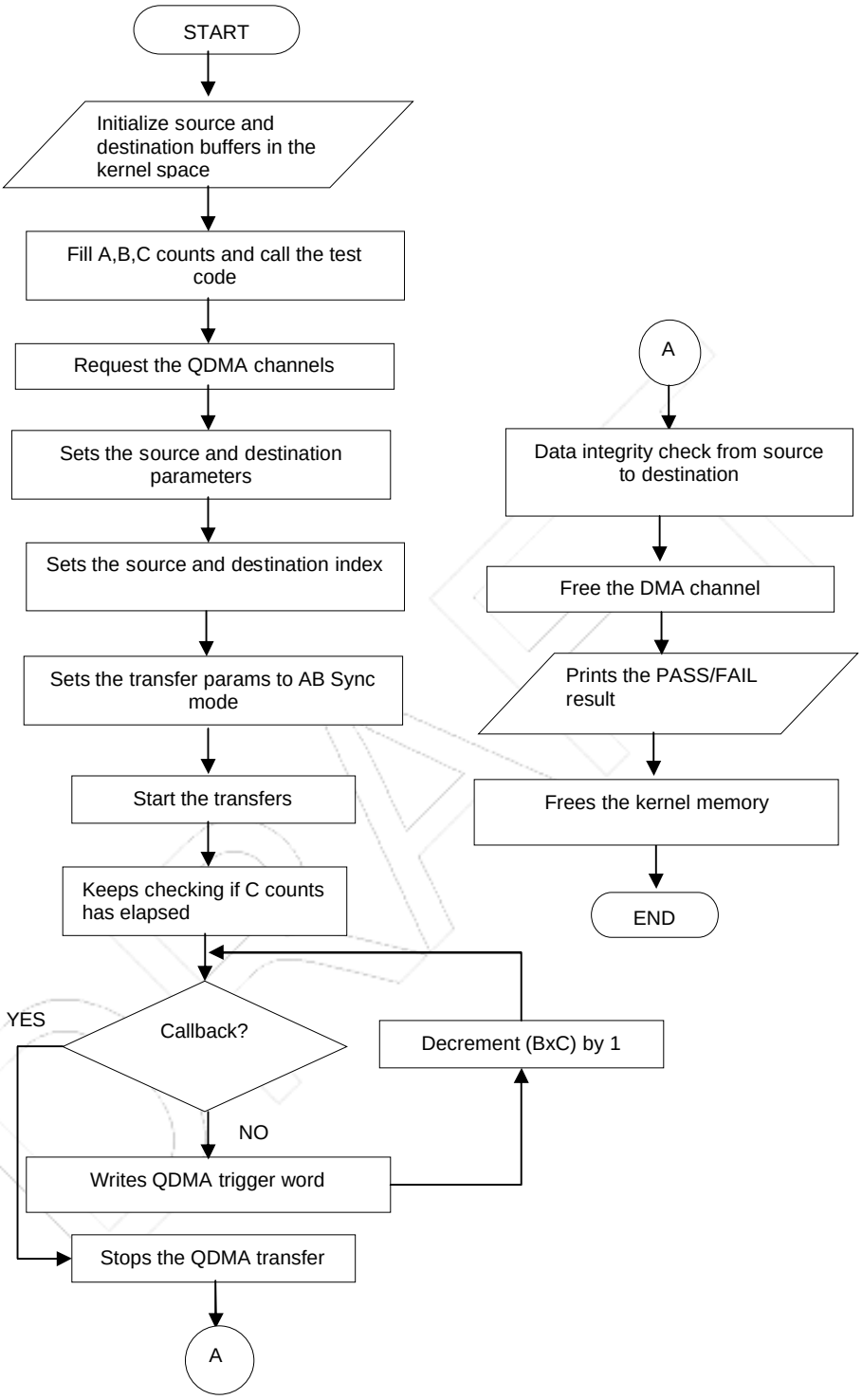


Figure 4-32. Flow Chart of QDMA AB Sync Incremental Mode

4.9 VLYNQ Performance Tests

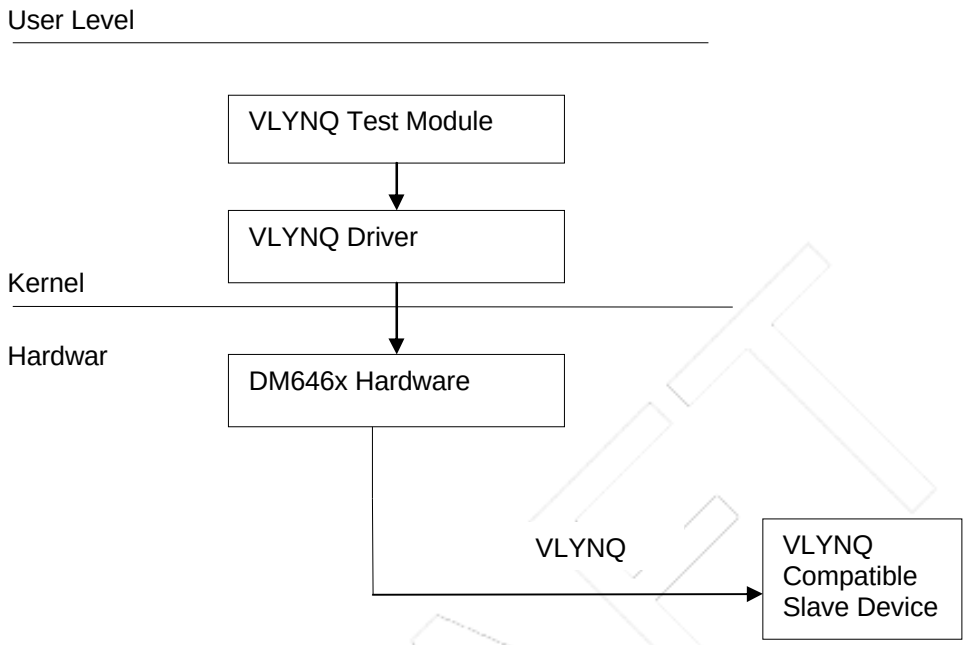


Figure 4-33. VLYNQ Data Flow Diagram

4.9.1 CPU Transfer:

The VLYNQ performance with CPU transfer is measured as time taken in micro seconds for transferring various data sizes to/from VLYNQ space through CPU.

Figure 4-34 shows the flow chart for the function `vlynq_cpu_transfer_perf_test ()`.

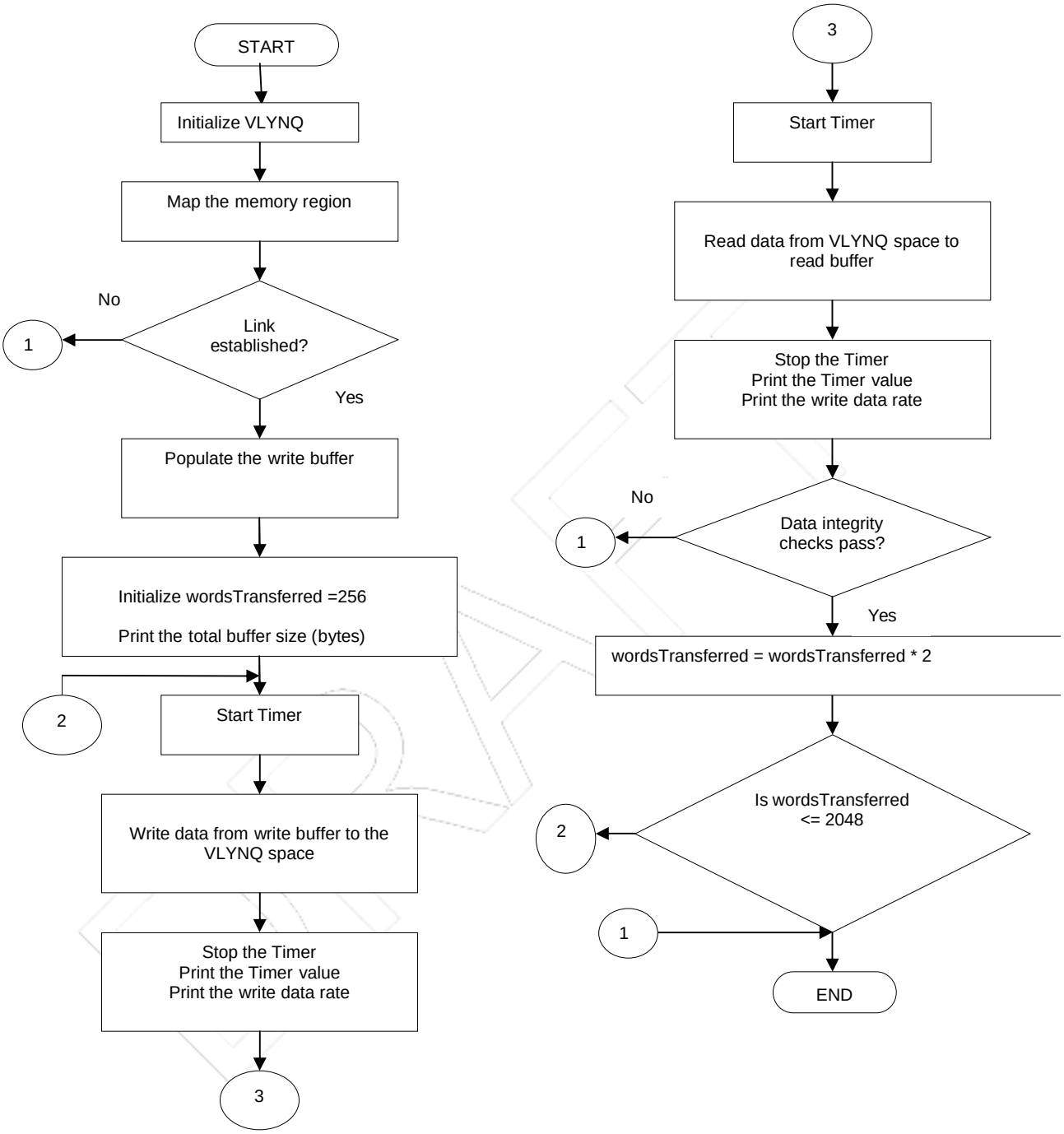


Figure 4-34. Flow Chart of VLYNQ CPU Transfer

4.9.2 EDMA Transfer:

The VLYNQ performance with EDMA transfer is measured as time taken in micro seconds for transferring various data sizes to/from VLYNQ space through EDMA.

EDMA Transfer modes:

- ASYNC
- ABSYNC

Note:

The EDMA_ASYNC_TRANSFER should be defined, if performance value in ASYNC EDMA mode is needed. Currently, performance value in ABSYNC mode only is provided as output.

Figure 4-35 shows the flow chart for the function `vlynq_edma_transfer_perf_test ()`.

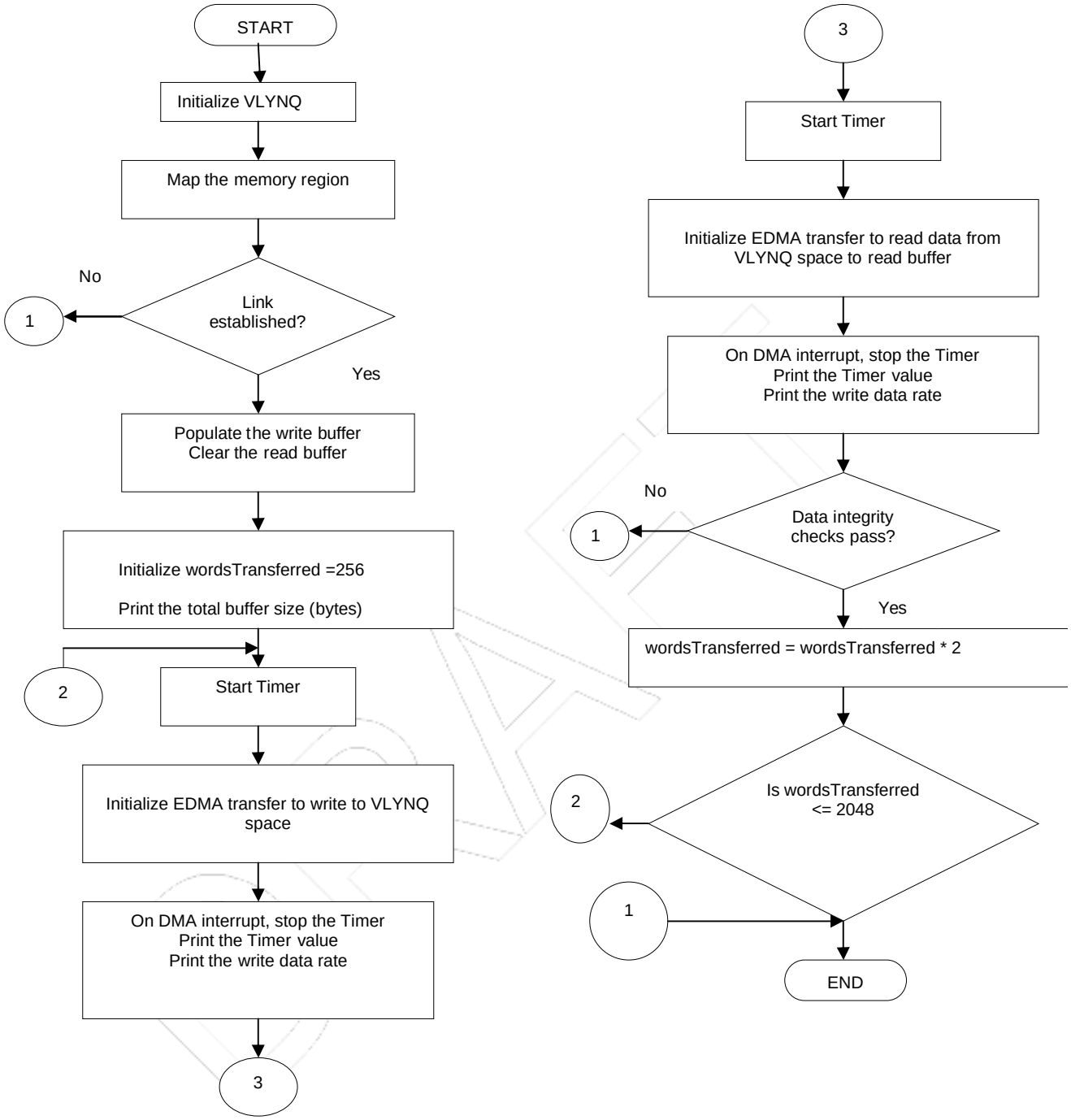


Figure 4-35. Flow Chart of VLYNQ EDMA Transfer

Appendix A

CPU Load Measurement

This appendix provides details about procedure followed for measuring CPU Load and recommended conditions for running CPU Load measurement.

A.1 /proc/stat

The proc file system acts as an interface to internal data structures in the kernel. It can be used to obtain information about the system. Various pieces of information about kernel activity are available in the /proc/stat file. All of the numbers reported in this file are aggregates since the system first booted. The parameters reported by /proc/stat are listed in the below section

A.1.1 /proc/stat parameters

1. User Level Parameters
    a. user - normal processes executing in user mode
    b. user_rt - normal processes executing in user mode with real time priority
    c. nice - niced processes executing in user mode
2. System Level Parameters
    a. system - processes executing in kernel mode
    b. system_rt - processes executing in kernel mode with real time priority
    c. irq - servicing hardware interrupts
    d. softirq - servicing softirqs
    e. iowait - waiting for I/O to complete
    f. idle - twiddling thumbs
    g. steal time – percentage of time a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor.

A.2 Procedure followed for calculating the CPU Load

All the parameters are read through '/proc/stat' before IO is started. Readings will be taken again after finishing IO. Difference between the two readings will give the amount of times spent for the IO to happen.

```
totalTime = user + nice + system + idle + iowait + irq + softirq
                + steal + user_rt + system_rt

idleTime = idle;
/* IOwait should not be accounted in CPU perspective, refer:
http://kbase.redhat.com/faq/FAQ_80_5637.shtm */
idleTime += iowait;

cpuLoad   = ((totalTime - idleTime) * 100) / totalTime;
```

A.3 Limitations of /proc/stat

1. The figures that are displayed are updated every jiffy (10 milli seconds). If the value of irq is 0 then it does not imply that the system had not been executing IRQs. It means that when timer interrupt is sampled during the last timer tick (every jiffy) it is found that the CPU was involved in an activity (for eg IO wait). So added the time (roughly 10ms) to the count of that particular activity (cpu_usage_stat.iowait). This issue can be solved by using an application/utility at kernel level to get the cpu status for every 1 milli second using timer interrupts, so that more accurate results can be obtained. However, getting the CPU status very frequently using the interrupts itself might cause a CPU Load, which is unintentional.

2. The test environment has to be controlled. The assumption here is that only the test process is actually running on the system along with the kernel processes running for that test process.

A.4 Recommended procedure while using /proc/stat to calculate CPU Load

1. To get the better average CPU Load run the tests for 100*10 msec (= 1 sec).

2. Do not run any other processes on system other that your particular test when measuring the CPU Load.

3. Do a calibration check to see if you are getting 0% Load before your test.

To get the average CPU Load for period of time, a utility / application is required which can read the /proc/stat and get the time spend by CPU in all the states (cpuStatusStart). After the period of time or completion of a task for which CPU Load needs to be calculated, read the /proc/stat again and get the time spend by CPU in all state (cpuStatusEnd).

Get the difference of time for respective states of CPU (cpuStatusDiff).
cpuStatusDiff.user = cpuStatusEnd.user - cpuStatusStart.user;
cpuStatusDiff.nice = cpuStatusEnd.nice - cpuStatusStart.nice;
cpuStatusDiff.system = cpuStatusEnd.system - cpuStatusStart.system;
cpuStatusDiff.idle = cpuStatusEnd.idle - cpuStatusStart.idle;
cpuStatusDiff.iowait = cpuStatusEnd.iowait - cpuStatusStart.iowait;
cpuStatusDiff.irq = cpuStatusEnd.irq - cpuStatusStart.irq;
cpuStatusDiff.softirq = cpuStatusEnd.softirq - cpuStatusStart.softirq;
cpuStatusDiff.steal = cpuStatusEnd.steal - cpuStatusStart.steal;
cpuStatusDiff.user_rt = cpuStatusEnd.user_rt - cpuStatusStart.user_rt;
cpuStatusDiff.system_rt = cpuStatusEnd.system_rt - cpuStatusStart.system_rt;

Calculate the CPU Load as mention in section “A.2 Procedure followed for calculating CPU Load”.

A.5 Enabling Debug mode for CPU Load

By default the test bench is not compiled in Debug mode for CPU Load. To enable the debug mode add the below line to GENDEFS under psp_test_bench folder.

DEBUG = -DCPULOAD_DEBUG

This mode will give all the details on /proc/stat parameters.