Test Framework Enhancement and Automation Support for ST Media Framework

Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology
in
Electronics & Communication Engineering
(Communication Engineering)

By

PREETI DEWANI
(12MECC37)

Electronics & Communication Branch

Electrical Engineering Department

Institute of Technology

Nirma University, Ahmedabad-382 481

May 2014

Test Framework Enhancement and Automation Support for ST Media Framework

Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

in

Electronics & Communication Engineering

(Communication Engineering)

By

PREETI DEWANI

(12MECC37)

Under the Guidance of

Mr. Pavan Kumar Goyal	**Dr. D. K. Kothari**
Technical Leader, STM	**Professor, EC, IT, NU**

Electronics & Communication Branch

Electrical Engineering Department

Institute of Technology

Nirma University, Ahmedabad-382 481

May 2014

Declaration

This is to certify that

i The thesis comprises my original work towards the degree of Master of Technology in Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.

ii Due acknowledgement has been made in the text to all other material used.

Preeti Dewani

Certificate

This is to certify that the Major Project entitled **"Test Framework Enhancement and Automation Support for ST Media Frameowrk"** submitted by **Preeti Dewani (12MECC37)**, towards the partial fulfillment of the requirements for the degree of **Master of Technology in Communication Engineering** of Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this thesis, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date : **Place :** Ahmedabad

Guide :
Dr. D. K. Kothari
(Professor,EC)

Program Coordinator:
Dr. D. K Kothari
(Professor and section head,EC)

HOD,EE :
Dr. P. N. Tekwani
(Professor,EE)

Director
Dr. K Kotecha
(Director, IT, NU)

Acknowledgements

I offer my sincere thanks to the very individual who played their possible role in inspiring me, motivating me and helping me during the project.

I am deeply indebted to my guide Dr. D. K. Kothari (Professor and Program Coordinator, M.Tech - EC (Communication Engg.)) and industrial guide Mr. Pavan Kumar Goyal, Technical Leader, STMicroelectronics for their constant guidance, motivation and support. They have devoted significant amount of valuable time to plan and discuss the thesis work. Without their experience and insights, it would have been very difficult to do quality work.

I would like to express my gratitude and sincere thanks to Dr. P. N. Tekwani, Head Of Electrical Engineering Department and Dr. D. K. Kothari, Coordinator, M. Tech Communication Engineering program for allowing me to undertake this internship.

I also wish to thank Mr. Shivdeep Singh for their help, support and guidance. Without their guidance and support, it would have been very difficult to complete the project.

I wish to thank my friends for their delightful company which kept me in good humor throughout the year.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Preeti Dewani
(12MECC37)

Abstract

In Set Top Boxes new applications support are coming day by day. In windows it is easy to support Media applications but when it comes to open source community then there is a need of a multimedia framework which provide libraries to support applications developed on top of it. Applications are linked to plugins, provided by Gstreamer, to provide appropriate functionality.

There are many tools provided by Gstreamer. Some of them are gst-apps, gst-inspect, gst-discoverer, gst-launch which are useful for different purposes. For example, gst-launch is used for debugging purposes to create pipeline manually by connecting all elements, likewise gst-inspect is used to get detail about a particular element etc. With implementation of all these tools, human efforts can be reduced. So, a study of all these tools is required.

For each and every functionality developed using gst-apps, there is need to do regression testing to check that in built functionalities are not breaking which were developed earlier. To do so, some framework are designed using XML and PYTHON scripting, which are suitable to complete whole test procedure.

DVR i.e. Digital Video Recording is becoming a primary requirement of user. A continuous incorporation of new features is necessary in DVR and different use cases must be tested properly. Circular Buffer is one of the new implementation of DVR by which we can see latest buffer size recording. With new improvements there comes new issues which need to be resolved.

To do all such sort of testing we require to automate the test framework so that much of our time should not be wasted in doing all these. For this purpose shell and python scripting is used. Even GUIs are also prepared for same purposes.

For all that different applications are built using gstreamer and efforts are done in preparing gui for making work of user easy. Along with that bugs are solved in order to improve already built in functionality, circular buffer tool, bugs and testing is specially covered as an important part.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

There are many sources of building an application in Windows environment but there are problems with the development of application in open source environment. Gstreamer is the tool to develop the application in open source environment. As it is open to all as a result, new plugins and rpms(Red Hat Packet Manager) are coming day by day and support for new features are incorporated every minute. So, there is a need to do regression testing on the already developed code. As application is very generic to all types of users. Because of this testing is done for all types of sources in a different manner.
As it is difficult to do testing on all types of use cases, So, its automation is a necessary step to be followed. In Set Top Boxes FEC is used as an error correction technique but it has some limitations. It is not suitable to use it in case of large packet length and high packet loss rate. There is a need to use some new technique in worst environment conditions. Multiple Description Coding is the solution. MDC and FEC solves the problem all together.

1.2 Background:Introduction to STB(Set Top Boxes)

A Set Top Box (STB) or Set Top Unit (STU) is a device that connects to a television and an external source of signal, turning the signal into content which is then displayed on the television screen[1].

A set-top box is a computerized device that processes digital information. Set-top boxes (STB) come in many forms and can have a variety of functions. Digital Media Adapters, Digital Media Receivers, Windows Media Extender and most video game consoles are also examples of set-top boxes. Currently the type of TV set-top box most widely used is one which receives encoded/compressed digital signals from the signal source (perhaps your cable or telco TV provider's headend) and decodes/decompresses those signals, converting them into analog signals that your analog (SDTV) television can understand. The STB accepts commands from the user (often via the use of remote devices such as a remote control) and transmits these commands back to the network operator through some sort of return path. Most set-top boxes deployed today have return path capability for two-way communication.

STBs can make it possible to receive and display TV signals, connect to networks, play games via a game console, surf the Internet, interact with Interactive Program Guides (IPGs), virtual channels, electronic storefronts, walled gardens, send e-mail, and videoconference.

Many STBs are able to communicate in real time with devices such as camcorders, DVD and CD players, portable media devices and music keyboards. Some have huge hard-drives and smart card slots to put your smart card into for purchases and identification.

1.3 ARCHITECTURE

A Set Top Box comprises of Front-End and Back-End which includes several major blocks: front-end, security, audio/video (A/V) decoding, memory, return path and digital interface for peripherals.

1.3.1 Front-End

The front-end block is comprised of a tuner and a demodulator (as shown in fig. 1.1). It translates a RF signal into a digital-corrected stream, also called the transport stream. The digital demodulator in the cable is QAM, QPSK for satellite, and COFDM for terrestrial connection. Front-end devices are now embedded into the tuner can. The integration of complete tuner functions on silicon is now available for satellites, while silicon tuner for cables is still under development. The front-end components are:

- Tuners: Tuner receives a digital signal from a network and tunes to a particular channel in the corresponding frequency range. Basic purpose of a tuner is to amplify, detect, select and convert a desired RF signal from an antenna (or cable) to a demodulator while keeping the signal quality as much as possible. Main features and qualities of a tuner are:
  - Sensitivity: defines the lowest RF signal power that can be received.
  - Selectivity: capability of the tuner to discriminate the desired signal out of all others received signals.
  - Dynamic range: span of acceptable of input desired signal power from the lowest to the highest.
  - Fidelity: capability to keep the desired signal characteristics.
- Three types of tuners are generally being used in STBs:
  - In-band tuners: The tuner is generally controlled by an I2C bus to select the required channel in the cable band (VHF/UHF from 50MHz to 860MHz), converts it into an IF, then feeds to a QAM demodulator.

  - Out-of-band tuners: They facilitate the transfer of data between the head-end systems and the STB. They are typically used in a cable box when providing interactive services, and operate within the 100MHz to 350MHz frequency band.
  - Return-path tuners: These tuners allow for activation of the return path and send data back to the head-end station. The frequency band allocated to the upstream is located between 5MHz to 65MHz frequency band.

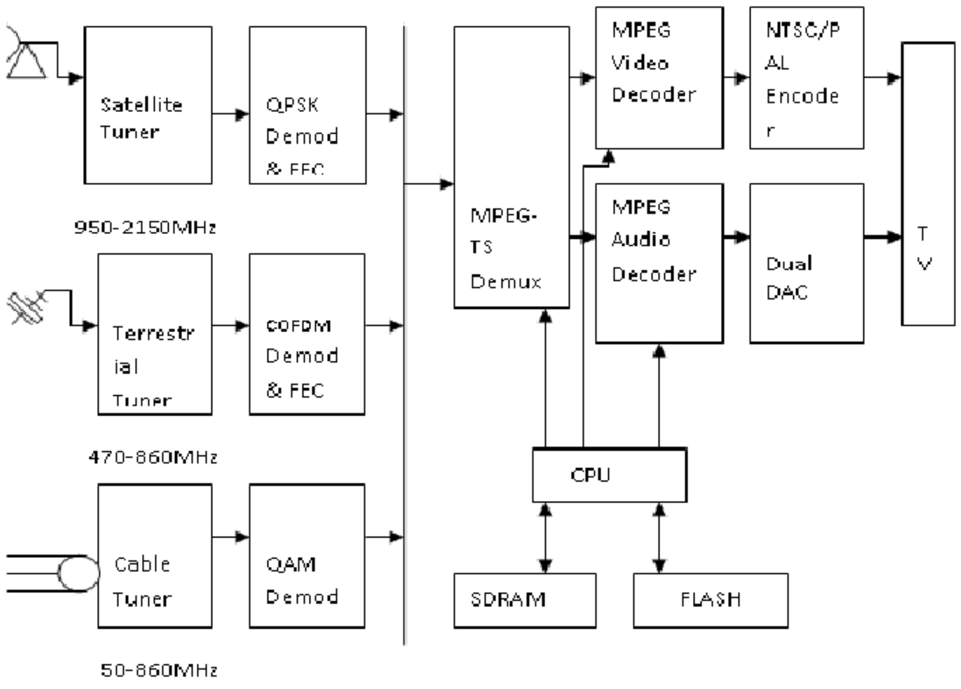


Figure 1.1: Set Top Box

- Demodulators, modulators: The baseband output signal from the tuner continues on to the QAM demodulator and FEC, which performs sampled IF to bit streams that are fully compliant with ITU-T J83 annexes A/B/C or DVB-C specification. These bit streams contain A/V and data in the backend demultiplexer for MPEG-2 block processing. The modulator is to reverse the demodulator's actions and use the STB to deliver a signal to the return-path tuner. Type of demodulation depends on the type of tuner being used as given below.

QPSK demodulator is used in case of satellite transmission.

OFDM demodulator is used in case of terrestrial transmission.

QAM demodulator is used in case of Cable transmission

Digital demodulator embedded into tuner is called Network Interface Module (NIM). NIM is controlled using I2C bus to select required channel, converts it into IF signal and feeds it to the demodulator.

1.3.2 Back-End

Back-end basically comprises of TS demultiplexer, A/V decoder, digital video encoder, DAC and CPU with on-site memory. These components are described as follows:

- Demultiplexer: Demultiplexer (demux) extracts all the useful information from the TS, since MPEG-2 data streams consist of a number of unique data packets and uses packet ID to identify each packet that contains data, A/V and interactive services. The demux examines every packet ID, selects packets, decrypts them, and then forwards them to their specific decoder.

- Decoders: Decoders are required to convert the digital bit stream back into the format accessible by the subscriber. The video decoder converts video packets into a sequence of pictures. Next, the audio bit stream is sent to the audio decoder for decompression so it can be sent to the speakers. Table formats are decompressed using data decoders. Then, the decoded data is presented to the set-top processor. Sometimes, JPEG and MP3 decoder are also embedded to process digital still picture and compressed music.

- CPUs and memory: CPUs initialize various STB hardware components, process Internet and interactive TV applications, manage hardware interruptions, pull data from memory and run programs. CPU in STB is typically a 32bit processor with speed ranges from 50MHz to 300MHz. It always contains an arithmetic logic unit, a control unit and a clock.

1.3.3 Conditional Access Module

STB also contains one more module called "Conditional Access (CA) Module". The security usually refers to the conditional access (CA) system. This is to protect the TV content against piracy. CA provides the security between the subscriber and the network service provider, and acts as a gateway allowing viewers to access digital services. The primary purpose of the CA system is to control the subscriber's access to pay services through DTV. A customer with a valid contract can access a particular service. All pay-TV programs are scrambled before transmission. From there, scrambled transport stream goes into the CA system for descramble processing (as shown in fig 1.2). Descrambling will be authorized according to the rights that will be stored on the Smartcard [1].

There are two kinds of CA system:

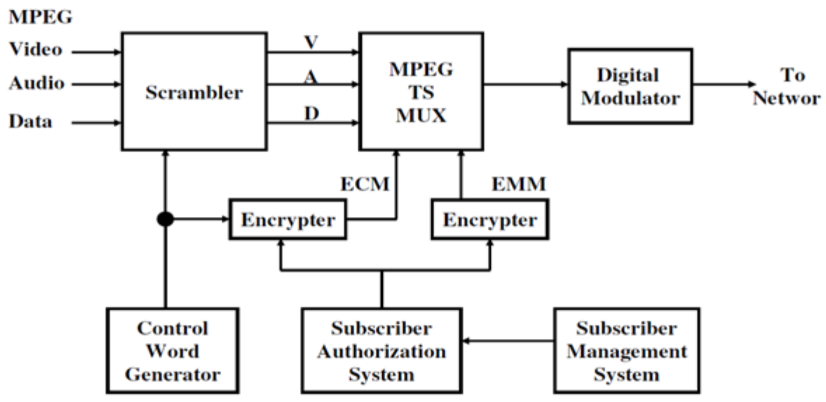


Figure 1.2: Conditional Access System1

- The Embedded CA in the STB and

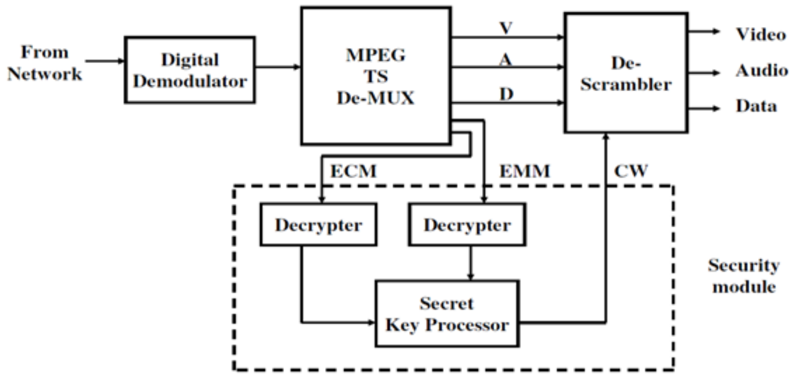


Figure 1.3: Condiditonal Access System

- The one located in the PCMCIA (Personal Computer Memory Card International Association).

Whenever an MPEG-2 TS carries encrypted (or scrambled) services, the TS also carries two types of messages called EMM (Entertainment Management Message) and ECM (Entitlement Control Message). An EMM carries a list of Pay TV services which the owner of that STB is entitled to view and also the date up to which he is entitled to receive them. The STB has an internal descrambler controlled by embedded conditional access software, which calculates the descrambler control words (CW) from the ECM (used by the 'descrambler' in the STB to descramble the picture and make it intelligible again) and keys contained in a subscriber smart card with valid access rights updated by the Entitlement Management Message (EMM). The other CA system, located on the PCMCIA module, is called the conditional access module (CAM). The interface to connect a CAM in an STB is called the Common Interface (CI) [1].

1.4 Categorization of Set Top Box depending on Transmission Media

1.4.1 Digital Satellite STB

Direct broadcast satellite or Direct-to-Home (DTH) is most popular to receive digital TV signal in most of the world. In Front-end tuner receives a digital signal from a satellite network and tunes to a particular channel in the frequency range of 950-2250MHz. The signal is digitally demodulated using QPSK demodulator to obtain digital data. Forward Error Correction (FEC) decoder will check errors and send TS to Back-end. Back-end TS demultiplexer extracts audio, video and clock and sends to A/V decoder and smart card.

1.4.2 Digital Terrestrial STB

It operates in UHF range from 470-860MHz and uses OFDM modulation to avoid multipath fading. It needs Yagi antenna to capture the signal. Its Front-end needs RF tuner, OFDM chip with FEC decoder. The tuner receives RF signal and performs channel selection by locking on to a given frequency and converts this signal into IF signal 36 MHz. Then OFDM demodulation is performed by applying 2K or 8K FFT to IF signal. If some carriers damaged due to multipath fading, lost bits are recovered by FEC decoder. FEC decoder and Back-end are same as in digital satellite STB.

1.4.3 Digital Cable STB

It operates in the VHF and UHF range from 50-860MHz. It uses QAM modulation since transmission on cable is hardly subjected to disturbances. 64- or 256-QAM permits 6 or 8 bits instead of 2 bits/symbol in QPSK used in satellite STB. Front-end needs RF tuner, QAM chip with FEC (R-S code) decoder. Tuner tunes one of the video channels in 50-860 MHz and selects channel by locking on to a given frequency and converts this signal into IF 36MHz. QAM demodulates IF signal into digital TS. Bit errors are corrected by FEC decoder. Symbol rate lies in the range from 0.87 to 11.7 Mbaud. Back-end is same as digital satellite STB.

1.4.4 Internet Protocol (IP)-STB

Video over IP allows TV viewers to see their favourite channels from say USA, Japan, Korea. IPTV will bring every channel in every Nation to every viewer. Increasing worldwide broadband penetration and efficient video compression H.264 enable Telecom services to deliver IP-video to customers. IP-STB consists of MPEG A/V decoder along with Ethernet controller. Ethernet controller implements MAC (Media Access Control) and PHY (Physical layer) portion of the CSMA/CD protocol at 10 and 100Mb/s. It integrates IEEE802.3 PHY for twisted pair Ethernet applications.

1.5 STB SOFTWARE

There are different software layers in a STB. The structural diagram of general STB software is shown in figure 1.4. An operating system (OS) is the most important piece of software in a STB. An OS is a suitable of programmes used to manage the resources in a STB. In particular it is the OS, which talks to the STB hardware and manage their functions such as scheduling real time tasks, managing limited memory resources, etc [8]. A STB OS is arranged in layers with each layer adding new capability. At the heart of any STB OS is the "Kernel" layer, which is stored in ROM. Once the STB is powered up, the kernel will be loaded first and remains in memory until the STB is powered down again. Typically the kernel is responsible for managing memory resources, real time applications and high-speed data transmission. The kernel supports multi threading and multi tasking which allows a STB to execute different sections of a program and different programmes simultaneously.

In addition to the kernel, a STB needs a 'loader' to enable the TV operator to upgrade 'resident applications' or download 'OS patches' to STB. A resident application is a program or a number of programs that are built into the memory of the STB.

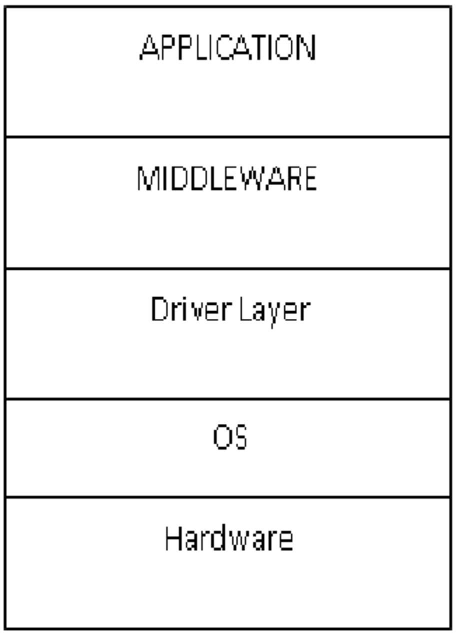


Figure 1.4: STB Software

The STB also requires 'drivers' to control the various hardware devices. Every hardware component in the STB must have a driver. A driver is a program that translates commands from the TV viewer to a format that is recognizable by the hardware device.
Finally a STB OS needs to incorporate a set of Application Programme Interfaces which are used by the programmers to write high-level applications for a specific API. An API is basically a set of building blocks used by software developers to write programs that are specific to a STB OS environment.

Central to the new software architecture of a STB is a connection layer that acts as communications bridge between the OS and the 'subscriber applications' called

'Middleware'.

Middleware is a relatively new term in the set top business. It represents the logical abstraction of the middle and upper layers of the communication software stack used in set top software and communication system. Middleware is used to isolate set top application programs from the details of the underlying hardware and network components. Thus set top applications can operate transparently across a network without having to be concerned with the underlying network protocols. This considerably reduces the complexity of content development because applications can be written to take advantage of a common API.

1.6 Organization of Report

This report comprises of nine chapters including this chapter. Chapter 1 comprises of motivation and general introduction to set top boxes. Chapter 2 include whole literature survey related to formation of stream, compression and types of modulation techniques. Chapter 3 include all hardware as well as software aspects of the environment. Chapter 4 consists of all necessary details required to run a test framework. Chapter 5 has all details related to DVR and its various usecases. Chapter 6 shows the implementation of circular buffer. Chapter 7 has detail about gstreamer, its tool and application designed using it. Chapter 8 include details of all guis designed for the purpose of automation. Chapter 9 comprises of conclusion and future scope.

Chapter 2

Literature Survey

2.1 Digital Era

Digital TV (DTV) is becoming an emerging consumer electronics appliance. Digital Television is the successor of analog TV and all broadcasting will be done in digital format. A small box sits on top of a standard TV set is called Set-top Box (STB). STB is central to this migration from analog-to-digital Broadcasting. STB will become a gateway to the digital information super highway For digital broadcasting, TV signal is

i. Digitized
ii. Compressed and
iii. Digitally modulated.

2.1.1 Digitization

Digitization of TV signal is carried out using Pulse code Modulation (PCM), shown in figure 2.1. PCM process needs

i. Sampling
ii. Quantization and
iii. Encoding

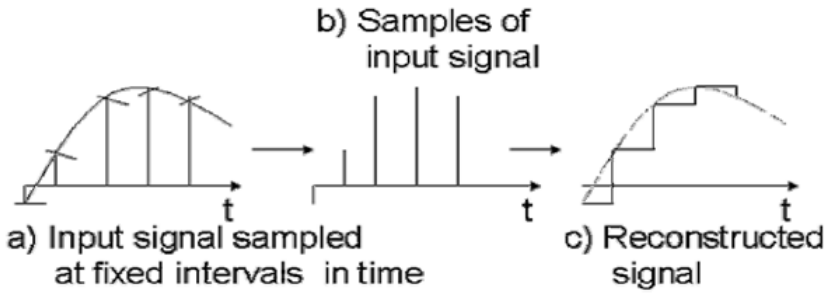

b) Samples of input signal
t
t
t
a) Input signal sampled at fixed intervals in time
c) Reconstructed signal

Figure 2.1: PCM Demonstration

- Sampling: Sampling determines samples corresponding to instantaneous amplitude of the input signal at uniform intervals
- Quantization: Input signal is divided into number of levels. Quantization allocates levels to the amplitude of sample values. Each sample peak falls within some specific level.
- Encoder: The value is translated into binary code using encoder.

2.1.2 COMPRESSION

Digitization strictly need to follow Nyquist Theorem to avoid the problem of aliasing. After Digitization:

- Video Compression: Display of TV frames rate from 50 to 60 pictures per sec [2]. is necessary. To provide sufficient visual information each picture needs to display 500 picture elements (pels) each in horizontal and vertical direction BW reduction of 2:1 achieved by dividing each frame into 2 fields containing half of total lines. Lines in first field are odd and in second field even called interlace scanning. As shown in figure 2.2, picture degradation due to interlace is known as flickering. Color TV picture requires Red, Green and Blue color

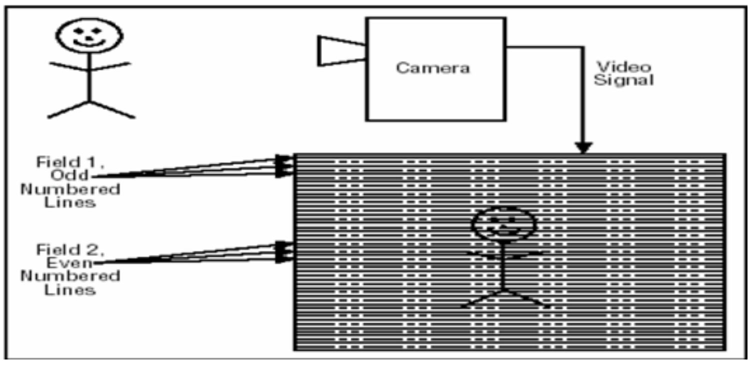


Figure 2.2: Flickering Due to Scanning

images. Interlacing could reduce to 15MHz. Three color TV signals are translated into 2 color difference signals R-Y and B-Y. Y is luminance (Brightness) = 0.3R+0.59G+0.11B, as shown in figure 2.3

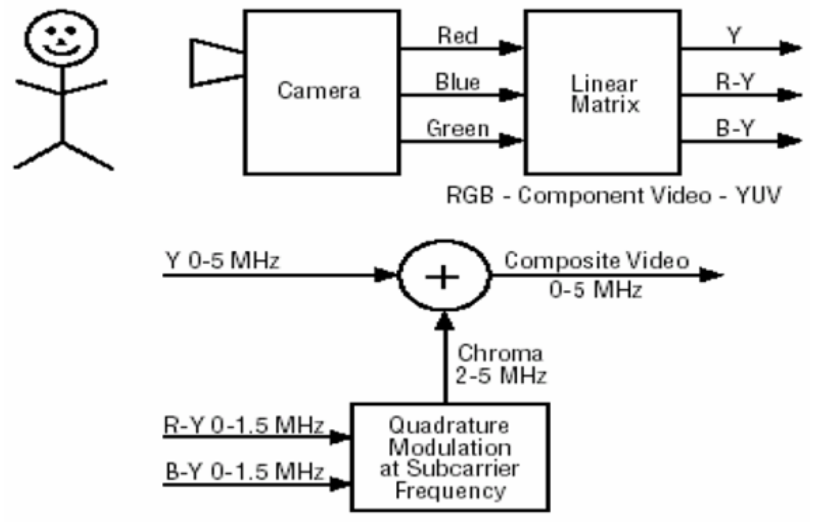


Figure 2.3: Composite Signal

- To do compreesiion certain points are to be considered:
  - TV picture information contains real information called Entropy and remaining information is Redundancy. In large plane areas of picture adja-

cent pels are correlated.

- Spatial Redundancy Coding: Line-to-line correlated called Spatial redundancy. Intra-frame Discrete Cosine Transform (DCT) coding is used to remove spatial redundancy.DCT is lossless reversible mathematical process, converts data from time domain to frequency domain. DCT performed over 8 pels X 8 lines (block) to produce DCT coefficients. Coefficients corresponding to zero frequency is called DC coefficient. DCT itself does not achieve any compression. Coefficients are weighted and truncated to get compression. Further data is compressed using Entropy coding. Entropy coding (Lossless) consists of Run Length Coding (RLC) transmit address and number of 0's or 1's for strings of 0's and 1's Other Entropy coding is Variable Length Coding(VLC).
  i. Some DCT coefficients occur very often and others occur seldom.
  ii. Frequently occurring values are converted to short code words and seldom occurring values with larger code word.

- Temporal Redundancy Coding: Frame-to-frame correlated called temporal Redundancy. Inter-frame coding or frame-to-frame coding transmits only differences between pictures.
  The picture difference in picture is further compressed using DCT coding. The decoder adds the picture difference into previous picture to obtain current picture. Due to channel noise introduced, the error in picture will propagate indefinitely. Error could be reduced by sending complete picture periodically called Intra-coded picture (I-picture).

- Motion Compensation: Picture difference data increases as motion increases in the picture. Picture difference could be reduced by measuring motion (represented by Motion Vector). Motion is measured by comparing Y-data of 2 successive pictures (as shown in figure 2.4). Motion Vector (MV) used to shift part of previous picture called Motion Compensation (MC).

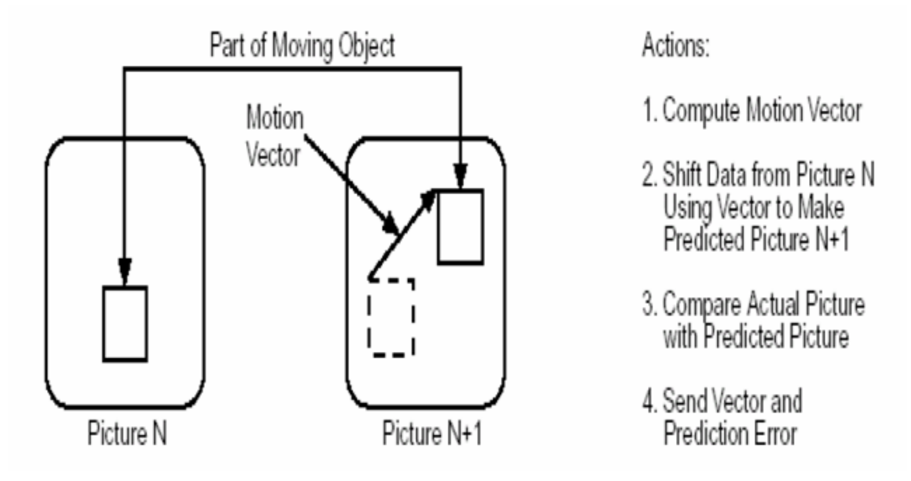

Part of Moving Object
Motion Vector
Picture N
Picture N+1
Actions:
1. Compute Motion Vector
2. Shift Data from Picture N Using Vector to Make Predicted Picture N+1
3. Compare Actual Picture with Predicted Picture
4. Send Vector and Prediction Error

Figure 2.4: Motion Vector ompensation

- The picture is divided into number of Macro Blocks (MB)

  i) In 4:4:4 sampling each MB contains four Y-blocks, four R-Y blocks and four B-Y blocks

  ii) In 4:2:2 sampling each MB contain four Y-blocks, two R-Y blocks and two B-Y blocks.

  iii) In 4:2:0 sampling each MB contains four Y-blocks, one R-Y block and one B-Y block.

- MPEG has 3 different types of pictures: I, P, B

  i) I-picture consists of DCT coefficients and no MV

  ii) P-picture is forward predicted from an earlier picture which could be I- or P-picture. P-picture has MV in addition to DCT coefficients, require half of data of I-picture (as shown in figure 2.5 and 2.6).

  iii) B-pictures are bi-directionally predicted from earlier and/or later I- and/or P-picture. B-picture has two MVs in addition to DCT coefficients require one-fourth of data of I-picture.

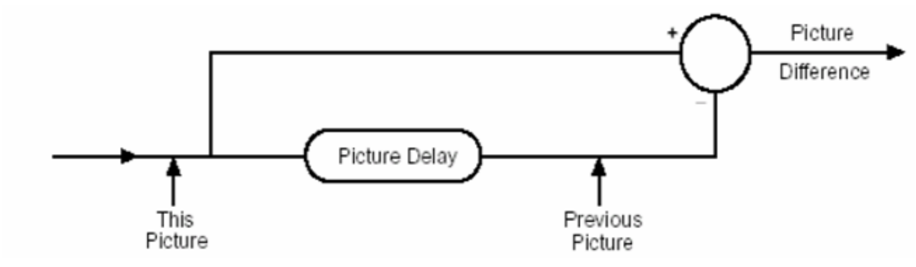


Figure 2.5: Prediction Scheme

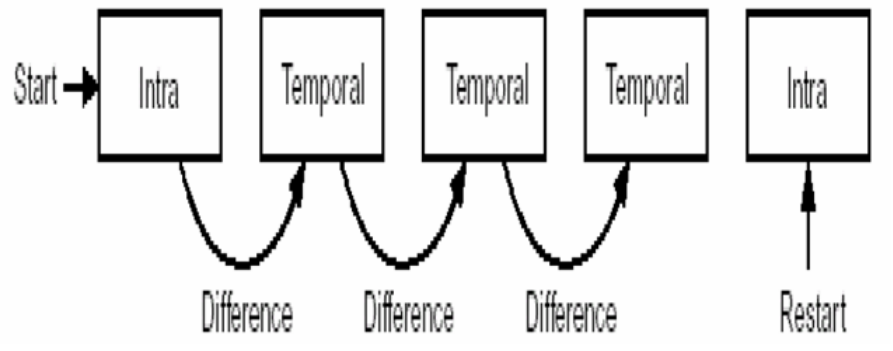


Figure 2.6: Prediction Scheme

- Slice is made up of several contiguous MBs [3], horizontal strips of picture from left to right. Picture consists of group of slices. Group of Pictures (GoP) is made up of a sequence of pictures starts with I-picture (as shown in figure 2.7). Video sequence includes one or more GoPs.

- Packetized Elementary Stream : An endless ES is divided into number of packets of convenient size (64kByte) for the application (as shown in figure 2.8).
  - Packets identified by Header that contains time stamps for synchronization
    DTS: When picture must be decoded.
    PTS: When picture must be presented to the decoder Output.
    * I-picture decoded and displayed to follow B-picture.
    * B-picture decoded and presented simultaneously (no PTS required).

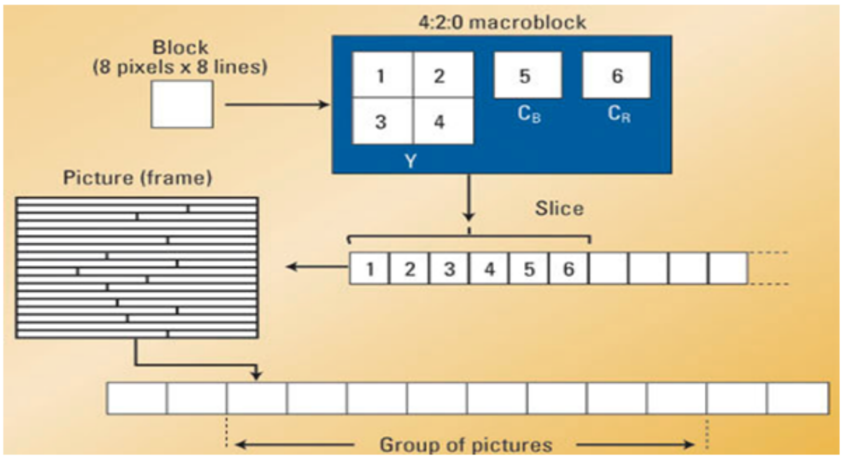


Figure 2.7: Formation of Picture

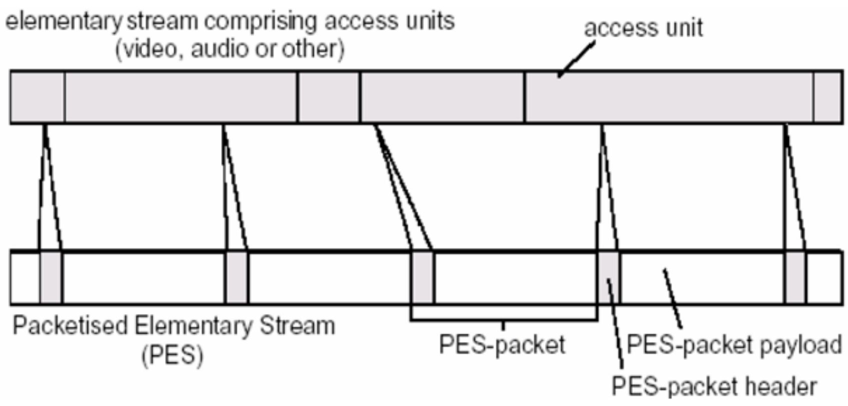


Figure 2.8: Elementary Stream

∗ P-picture separated by 3 picture periods.

∗ P-picture required both DTS and PTS.

- Program Stream (PS): As shown in figure 2.9, Several PES combined to obtain PS Works well on a single program with variable bit rate

- Transport Stream (TS): Works well with multiple program in a fixed bit rate . TS uses Program Clock Reference (PCR). With 4 bytes Header and 184 bytes payload (as shown in figure 2.10).

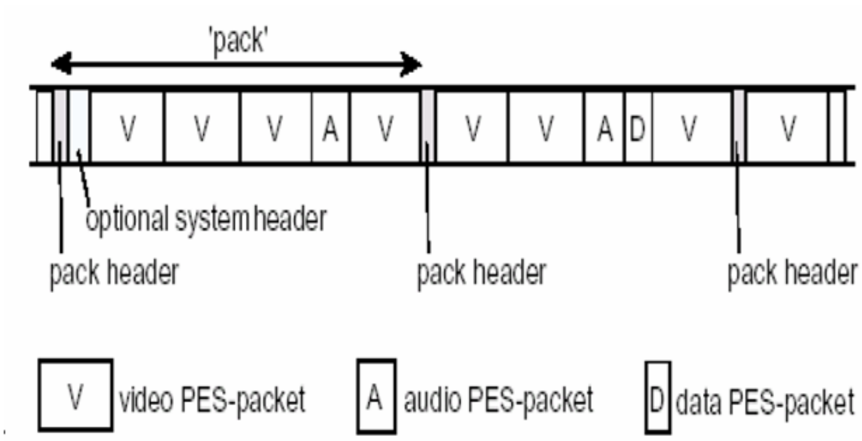


Figure 2.9: Program stream

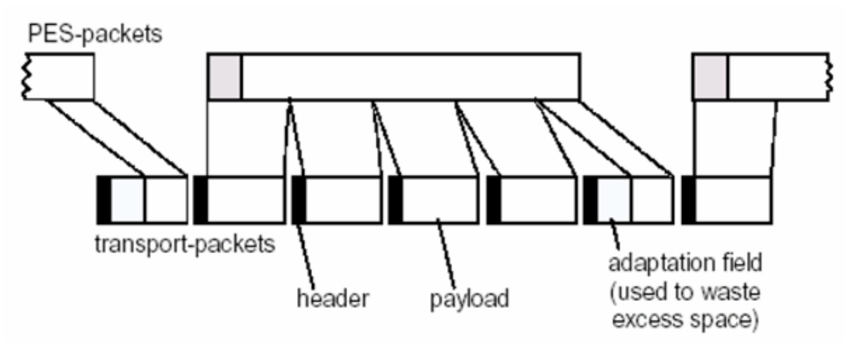


Figure 2.10: Transport Stream

- Program Specific Information (PSI)
  - In TS each transport packet is tagged with an appropriate Packet ID (PID) value indicating to which ES its payload belongs
  - There can be many ES comprising many different programs. Additional information required for decoder is called PSI and is present in every Transport Stream. PSI contains 4 types of Tables PAT, PMT, CAT, NIT, shown in figure 2.11
  - As shown in figure 2.12, Program Associated Table (PAT) PID=0 lists all programs available in TS. Conditional Access Table (CAT) PID=1 present

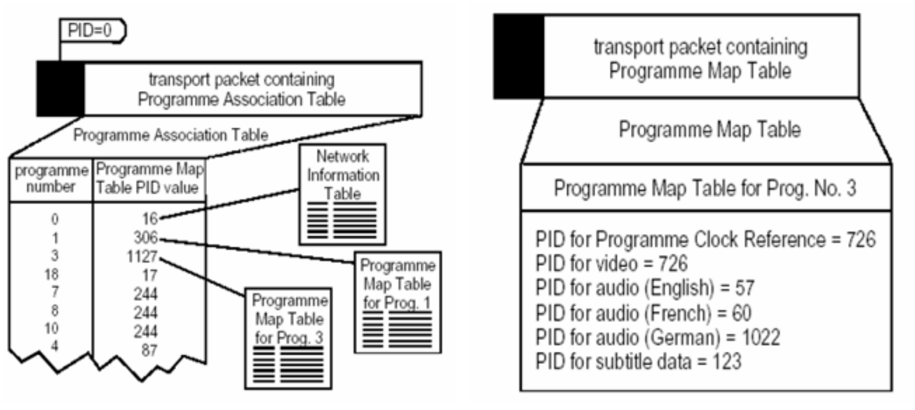

PID=0
transport packet containing Programme Association Table
Programme Association Table
programme number
Programme Map Table PID value
0 1 3 18 7 8 10 4
16 306 1127 17 244 244 244 87
Network Information Table
Programme Map Table for Prog. 1
Programme Map Table for Prog. 3
transport packet containing Programme Map Table
Programme Map Table
Programme Map Table for Prog. No. 3
PID for Programme Clock Reference = 726
PID for video = 726
PID for audio (English) = 57
PID for audio (French) = 60
PID for audio (German) = 1022
PID for subtitle data = 123

Figure 2.11: Program Specific Information

if any ES within TS is encrypted.

- Program number 0 points to Network Information Table (NIT), provides information about physical network e.g. which satellite transponder, which channel tuned, service name, modulation type etc.

2.1.3 MODULATION

- MODULATION : Modulation is the process of facilitating the transfer of information over a medium. Digital information to be transmitted via microwave, satellite or cable.
  - Digital modulator accepts a serial data stream and converts into analog format. Many different types of digital modulation:
    * Amplitude Shift Keying (ASK)
    * Frequency Shift Keying (FSK)
    * Binary Phase Shift Keying (BPSK)
    * Quadrature Phase Shift Keying (QPSK)
    * Quadrature Amplitude Modulation (QAM)
- Satellite Transmission-QPSK Modulation: Satellite transmissions have a few

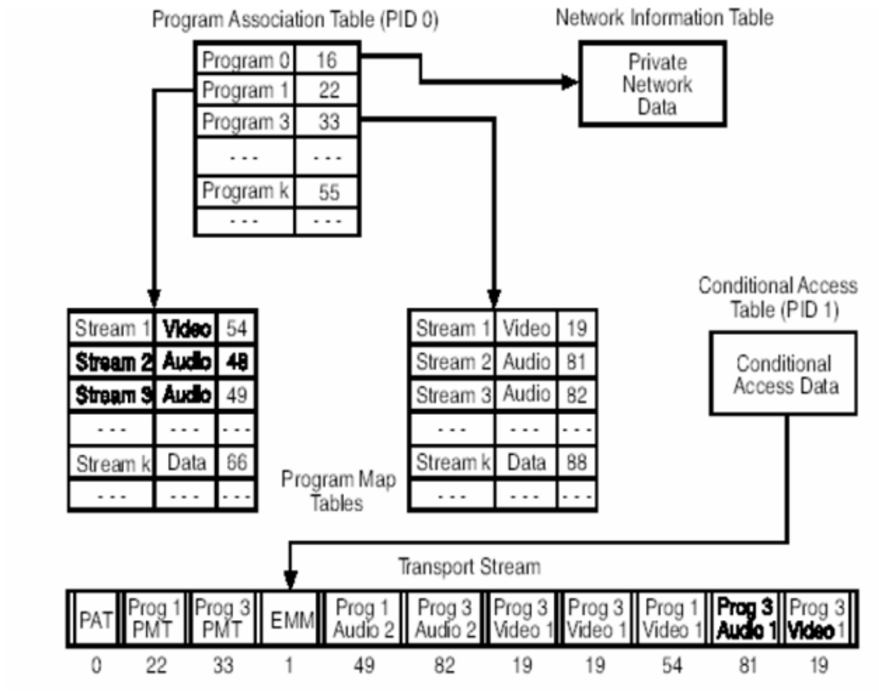


Figure 2.12: Details of PAT, PMT, NIT, CAT

unique characteristics. The signal has to travel an extremely large distance (36,000 kilometers) from the ground to the satellite and then another similar distance back to the earth.

– The satellite transmission is subjected to a broadband noise which is practically uniform at all frequencies.

– Since multiple channels are broadcast from the same satellite, the modulation technique should not be prone to Inter Channel interference.

– A satellite transponder has a fairly large bandwidth. Full transponders often have a bandwidth of 72 MHz . This is fairly a wide bandwidth, particularly when compared with the 7 or 8 MHz allotted to a channel on a cable systems. Hence a Digital Modulation technique used for Satellite Broadcasting (DVB-S) can use a fairly large bandwidth but should be capable of preserving the signal and maintaining a low Bit Error Rate

    (BER) even for very low signal strength.The QPSK Modulation system provides an ideal solution for this.

  - The word Quadrature simply means - Out of Phase by 90 Degrees. QPSK provides for 4 different states or possibilities for encoding a Digital Bit. This is because 2 components are used - one In Phase (I) and the other Out of phase or Quadrature (Q). This doubles the number of possible variations, from 2 to 4, that simple PSK offers. The QPSK system is now universally used, for all satellite DVB broadcasts.

- CATV Transmission - Q.A.M : Quadrature Amplitude Modulation (QAM) systems utilize changes of both, Phase Shift Keying and Amplitude Shift Keying to increase the number of states per symbol. Each state is defined with a specific variation of both - Amplitude AND Phase. This means that the generation and detection of symbols is more complex than a simple phase detection as in QPSK employed for Satellite Transmissions (DVB-S) because in Q.A.M. the Amplitude changes have also to be detected.

  QAM modulation is ideal for use in CATV networks. A cable system provides different transmission characteristics compared to satellite transmissions.

  - The bandwidth allocated per channel is restricted - just 6 to 8 MHz. Hence the Digital Modulation system must densely pack the digital data in a small bandwidth (unlike a satellite based transmission).
  - The signal levels are significantly higher than for satellite transmissions. Since the Carrier (signal strength) is larger, the Carrier to Noise (C/N) ratio is always fairly good in a CATV network.
  - A large number of channels are modulated and carried simultaneously on the same cable. Hence the modulation scheme should provide good Inter Channel Interference suppression.

  QAM comfortably meets all these requirements.

  - Since the Phase and Amplitude are varied in QAM Modulation, a large number of states or possible discreet values can be created to provide dense Digital Modulation.
  - Each time the number of states or options per symbol is increased, the bandwidth efficiency also increases. This bandwidth efficiency is measured in bits per second/Hz. As higher density modulation schemes are adopted, the Decoder or Demodulator gets progressively more complex.

- Terrestrial Transmission - O.F.D.M.: OFDM causes less interference to analog transmissions than an analog signal would, because it doesn't have the same strong carrier and subcarrier elements. Also, because there is a specific spacing between carriers of the same phase (guard interval), the signal is immune to multi path reflections or " Ghosts ".

  - The biggest concern for proper reception of terrestrial broadcast is multi path distortion, or " Ghosts ". This happens when a signal arrives at the receiving antenna from multiple paths or direction. These multiple signals add up at the antenna, creating multiple images or "Ghosts" on the TV screen. Analog transmissions cannot prevent "Ghosts". Terrestrially transmitted Television signal should preferably not interfere with other terrestrial transmissions such as those for wireless radio etc.
  - Orthogonal Frequency Division Multiplexing (OFDM) is a type of Frequency Multiplexing. In Frequency Multiplexing, multiple carriers are used at different frequencies; each carrier is separated by an unused band of frequencies called a "Guard Band". A Digital Terrestrial transmission (DVB-T) for a single television channel can utilize up to 8000 separate carriers. Orthogonal here refers to a phase difference of 90 Degrees between two adjacent carriers. Using Orthogonal Frequency Division Multiplexing (OFDM) Modulation, 2 Adjacent Carriers will overlap without causing

any interference because the two carriers are out of phase by 90 degrees. The overlapping of carriers avoids wastage of frequency bandwidth.

2.2 Summary

This Chapter describes the formation of Transport Stream which is the stream received at the front end of Set Top Box. Its header contain all the relevant information required to run the program. Compression standard is also covered which specifies the need of compression, along with procedure adopted to obtain compression. Along with that types of modulation techniques adopted for set top boxes are also described.

Chapter 3

Setting Up The Environment

3.1 Hardware Aspects

Equipment needed: In order to work on Set top Boxes there is a need to set up an environment to work upon it. It include both software and hardware requirements. In order to set up an hardware environment, few things are required :

- Board supported by sdk2 (as shown in figure 3.1). It is a set top box provided by company.
- STMC2 i.e. ST microconnect which is used for a means of communication with the board.
- Network connection
- External USB HDD connected to the board. All required streams are there in it for different use cases. Even we can record a new file into it.
- In this hardware connection basically whole set up is done in order to communicate with set top box through desktop. Microconnect is connected with PC as well as board. Board is connected with PC via ethernet cables.

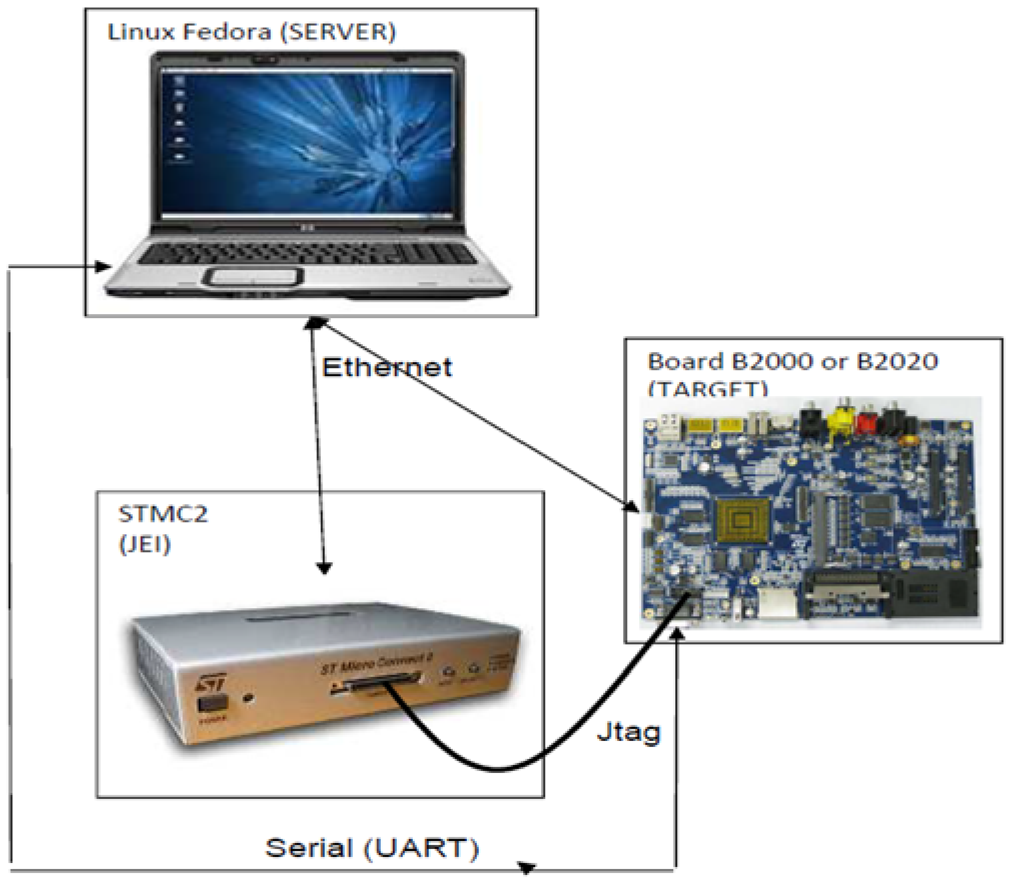


Figure 3.1: Hardware Set Up

3.2 Software Aspects

Software Development Kit(SDK) is the software stack which is used for whole assembly. This is the one on which all work is done. So, its environment should be set carefully otherwise even network issues can create lots of problem. Its prerequisites are:

- Step 1 : Exporting the Essentials

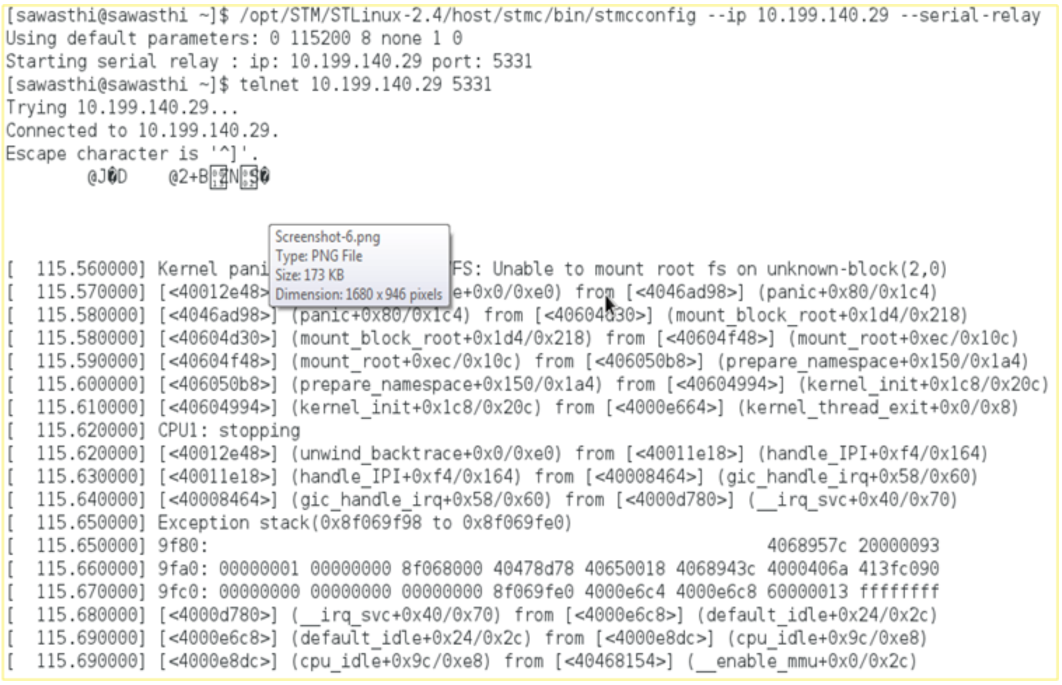
  Exporting the proxy:

  - export http-proxy = username: password: 8080
  - export https-proxy = username: password: 8080

- Step 2: Download the Repo
  - repo init -u tag-name -m sdk2-kernel3.xml –repo-url url-name”.
  - sync the new branches with the earlier one by;
    repo sync

  Step 3: Downloads and updates the rpms by
- - ./script-name I a armv7
  - Set the environment of the stack by
    ./script-name
- Step 4 : Building the Stack
  - make clean : to clean the already built modules
  - make all : to build the modules
- Starting the Application: Simulation 1
  - Start The sub shell that set up the environment
  - Go to the target build directory
  - Set the environment variable JEI, TARGET IP, SERVER IP,HW ADDR
  - Start Serial-relay(shown in figure 3.2)
- Manually : gst-apps
  - Step 1: Boot the board by
    make run (as shown in figure 3.3)
- Step 2: Start a telnet session by

  [root@sdk2-12] telnet TARGET IP (shown in figure 3.4)



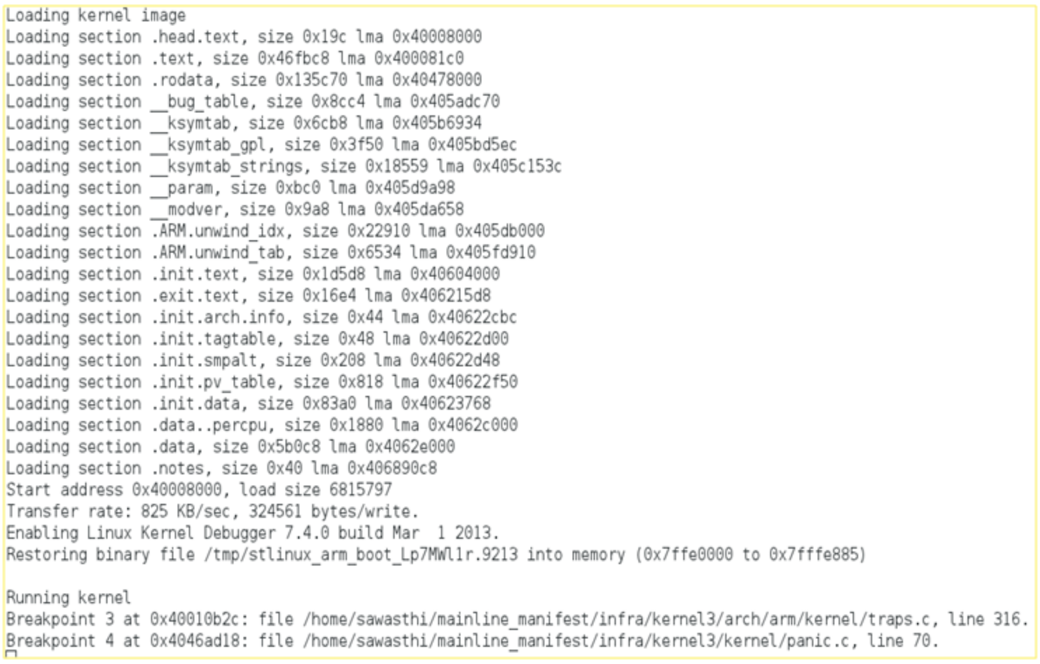
```
[sawasthi@sawasthi ~]$ /opt/STM/STLinux-2.4/host/stmc/bin/stmcconfig --ip 10.199.140.29 --serial-relay
Using default parameters: 0 115200 8 none 1 0
Starting serial relay : ip: 10.199.140.29 port: 5331
[sawasthi@sawasthi ~]$ telnet 10.199.140.29 5331
Trying 10.199.140.29...
Connected to 10.199.140.29.
Escape character is '^]'.
       @J�D    @2+B�ZN�S�



[  115.560000] Kernel pani                FS: Unable to mount root fs on unknown-block(2,0)
[  115.570000] [<40012e48>                e+0x0/0xe0) from [<4046ad98>] (panic+0x80/0x1c4)
[  115.580000] [<4046ad98>] (panic+0x80/0x1c4) from [<40604d30>] (mount_block_root+0x1d4/0x218)
[  115.580000] [<40604d30>] (mount_block_root+0x1d4/0x218) from [<40604f48>] (mount_root+0xec/0x10c)
[  115.590000] [<40604f48>] (mount_root+0xec/0x10c) from [<406050b8>] (prepare_namespace+0x150/0x1a4)
[  115.600000] [<406050b8>] (prepare_namespace+0x150/0x1a4) from [<40604994>] (kernel_init+0x1c8/0x20c)
[  115.610000] [<40604994>] (kernel_init+0x1c8/0x20c) from [<4000e664>] (kernel_thread_exit+0x0/0x8)
[  115.620000] CPU1: stopping
[  115.620000] [<40012e48>] (unwind_backtrace+0x0/0xe0) from [<40011e18>] (handle_IPI+0xf4/0x164)
[  115.630000] [<40011e18>] (handle_IPI+0xf4/0x164) from [<40008464>] (gic_handle_irq+0x58/0x60)
[  115.640000] [<40008464>] (gic_handle_irq+0x58/0x60) from [<4000d780>] (__irq_svc+0x40/0x70)
[  115.650000] Exception stack(0x8f069f98 to 0x8f069fe0)
[  115.650000] 9f80:                                                       4068957c 20000093
[  115.660000] 9fa0: 00000001 00000000 8f068000 40478d78 40650018 4068943c 4000406a 413fc090
[  115.670000] 9fc0: 00000000 00000000 00000000 8f069fe0 4000e6c4 4000e6c8 60000013 ffffffff
[  115.680000] [<4000d780>] (__irq_svc+0x40/0x70) from [<4000e6c8>] (default_idle+0x24/0x2c)
[  115.690000] [<4000e6c8>] (default_idle+0x24/0x2c) from [<4000e8dc>] (cpu_idle+0x9c/0xe8)
[  115.690000] [<4000e8dc>] (cpu_idle+0x9c/0xe8) from [<40468154>] (__enable_mmu+0x0/0x2c)
```

Figure 3.2: Serial Relay

- Step 3: Load the Modules
- Step 4: Start playing through gst-apps, as shown in figure 3.5

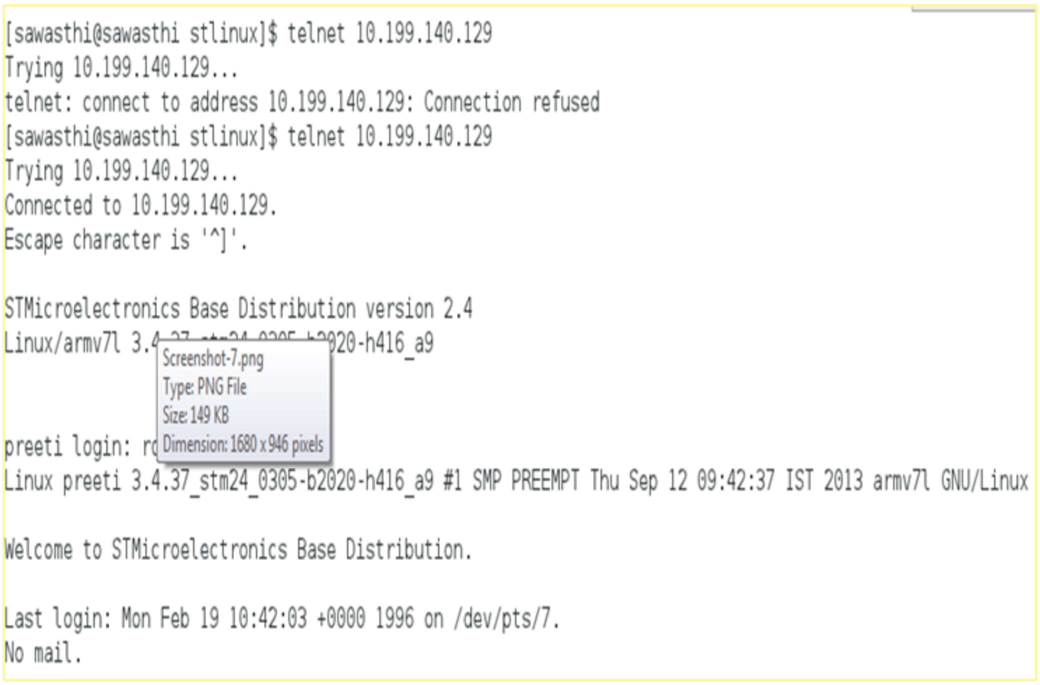
3.3 Summary

This chapter include prerequisites required to set up an environment. It include both hardware and software aspects. Along with that launching of command on target is also shown with by following some basic steps.



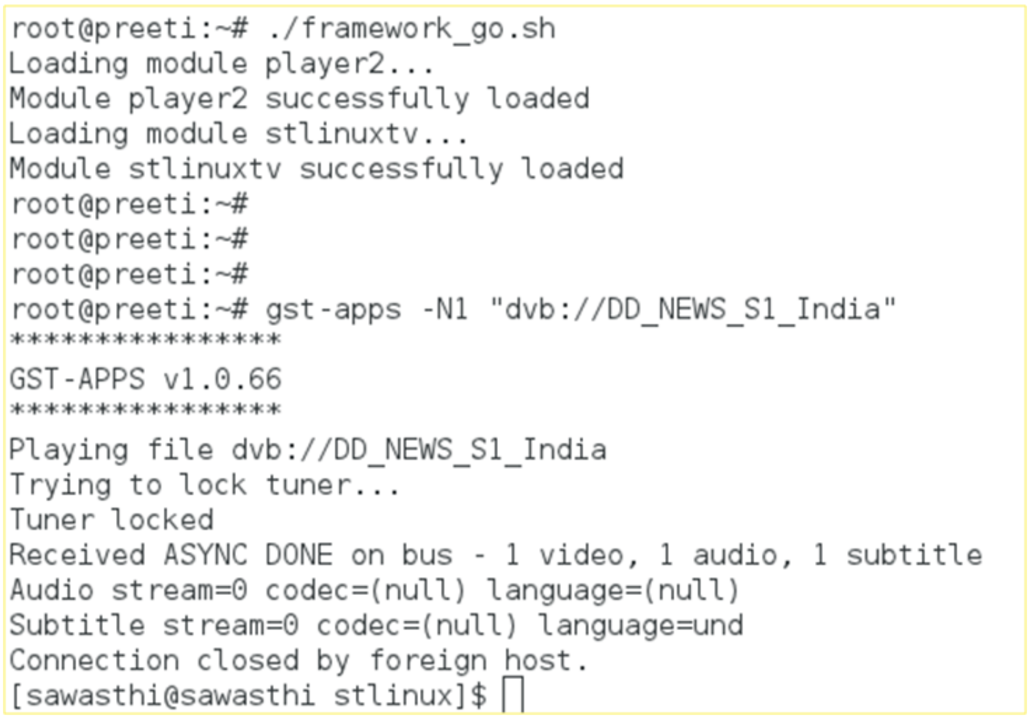
```
Loading kernel image
Loading section .head.text, size 0x19c lma 0x40008000
Loading section .text, size 0x46fbc8 lma 0x400081c0
Loading section .rodata, size 0x135c70 lma 0x40478000
Loading section __bug_table, size 0x8cc4 lma 0x405adc70
Loading section __ksymtab, size 0x6cb8 lma 0x405b6934
Loading section __ksymtab_gpl, size 0x3f50 lma 0x405bd5ec
Loading section __ksymtab_strings, size 0x18559 lma 0x405c153c
Loading section __param, size 0xbc0 lma 0x405d9a98
Loading section __modver, size 0x9a8 lma 0x405da658
Loading section .ARM.unwind_idx, size 0x22910 lma 0x405db000
Loading section .ARM.unwind_tab, size 0x6534 lma 0x405fd910
Loading section .init.text, size 0x1d5d8 lma 0x40604000
Loading section .exit.text, size 0x16e4 lma 0x406215d8
Loading section .init.arch.info, size 0x44 lma 0x40622cbc
Loading section .init.tagtable, size 0x48 lma 0x40622d00
Loading section .init.smpalt, size 0x208 lma 0x40622d48
Loading section .init.pv_table, size 0x818 lma 0x40622f50
Loading section .init.data, size 0x83a0 lma 0x40623768
Loading section .data..percpu, size 0x1880 lma 0x4062c000
Loading section .data, size 0x5b0c8 lma 0x4062e000
Loading section .notes, size 0x40 lma 0x406890c8
Start address 0x40008000, load size 6815797
Transfer rate: 825 KB/sec, 324561 bytes/write.
Enabling Linux Kernel Debugger 7.4.0 build Mar  1 2013.
Restoring binary file /tmp/stlinux_arm_boot_Lp7MWl1r.9213 into memory (0x7ffe0000 to 0x7fffe885)

Running kernel
Breakpoint 3 at 0x40010b2c: file /home/sawasthi/mainline_manifest/infra/kernel3/arch/arm/kernel/traps.c, line 316.
Breakpoint 4 at 0x4046ad18: file /home/sawasthi/mainline_manifest/infra/kernel3/kernel/panic.c, line 70.
```

Figure 3.3: Booting the Board



```
[sawasthi@sawasthi stlinux]$ telnet 10.199.140.129
Trying 10.199.140.129...
telnet: connect to address 10.199.140.129: Connection refused
[sawasthi@sawasthi stlinux]$ telnet 10.199.140.129
Trying 10.199.140.129...
Connected to 10.199.140.129.
Escape character is '^]'.

STMicroelectronics Base Distribution version 2.4
Linux/armv7l 3.4            20-h416_a9


preeti login: r
Linux preeti 3.4.37_stm24_0305-b2020-h416_a9 #1 SMP PREEMPT Thu Sep 12 09:42:37 IST 2013 armv7l GNU/Linux

Welcome to STMicroelectronics Base Distribution.

Last login: Mon Feb 19 10:42:03 +0000 1996 on /dev/pts/7.
No mail.
```

Figure 3.4: Connecting With Board



```
root@preeti:~# ./framework_go.sh
Loading module player2...
Module player2 successfully loaded
Loading module stlinuxtv...
Module stlinuxtv successfully loaded
root@preeti:~#
root@preeti:~#
root@preeti:~#
root@preeti:~# gst-apps -N1 "dvb://DD_NEWS_S1_India"
****************
GST-APPS v1.0.66
****************
Playing file dvb://DD_NEWS_S1_India
Trying to lock tuner...
Tuner locked
Received ASYNC DONE on bus - 1 video, 1 audio, 1 subtitle
Audio stream=0 codec=(null) language=(null)
Subtitle stream=0 codec=(null) language=und
Connection closed by foreign host.
[sawasthi@sawasthi stlinux]$
```

Figure 3.5: Running Application on Board

Chapter 4

Starting the Test Framework

4.1 Starting the Test Framework:

4.1.1 Prerequisite:

Once SDK2 is installed on our system and the application compiled, we can launch the Test Framework. At this point, we should have installed the SDK2 and the STLinux files, launched the sdk2.sh script and generated the SDK2 executable[4].

4.2 Test Framework Execution from the Command Line:

We can launch the Test Framework before or after the target has booted. COMMAND LINE OPTIONS General syntax:

- main.py [session options] rootfile.xml [targetIP]

4.2.1 The [session options]can be:

- initialboot

  Reboots the board once before launching the tests

- initialbootonly

  Reboots the board from the framework and exits

- listfiles

  Provides the list of parsed files for this test

- skipnn

  nn is an integer value, skips the first nn files

- startboard

  When running on board framework execution, apply Remote Shell Command tags from board configuration file

- stop-on-first-error

  For helping some debug cases, the framework exits on the first identified error, stops the streaming and provides the error code to the shell.

4.2.2 [root name]

Is the optional name we can encounter within a launch description of the root file. It helps launching only specific tests of a large program.

4.2.3 [additional path]

is here for extending the PATH search of the python and XML files within the Framework.

4.2.4 [target IP]

must be an ipv4 address.

4.3 Description:

The framework is made of data stored in XML format, and python code. General mind set is to maximize declarations and minimize code both on application side and in python interpreter side.

Declarations are made of:

4.3.1 Stream descriptions

This structure provides a unique file name, container, duration, codex for all tracks, resolutions, rates, stream ids. First parses directory tree and calls mediainfo for deep description of multimedia files. A second step is made of an xsl sheet transform that provides the compatible stream description XML required. We need a Mediainfo version installed in your Linux system for performing that import.

4.3.2 Stream locations

Aim is to make independent the way the file is accessed from the board and its properties. Role of Stream locations.xml is to tell depending on the modes (local, http), the Path in our system. Predefined given locations generally start with ”mounted”. To make this match on our board please ensure a ”mounted” directory exists under /root of the board[4].

4.3.3 Application abstraction

Set of applications is providing an abstraction layer, and information for the framework such as startup options for the launching command and interactive commands. Remark: The XML description of interactive commands gives the names of the functions to be called during tests.

- The command line options are bringing a number of options but need some python specific code for making proper use of them. This work is done today within module-dvbtest.py and module-gstapps.py.
- The interactive commands are simply storing node names that user script shall use for accessing abstracted names from application. Node content contains the application command.

4.3.4 Test cases, or Calls

This level is application independent. We should not take care of which application is running the test case (except testing advanced features only available on one application).

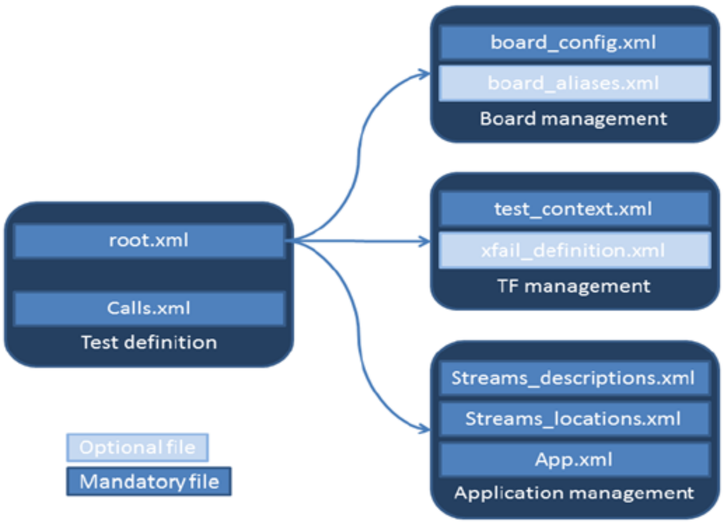


Figure 4.1: Structure of Calling files

4.4 Each test case defines:

A file selection. This let choose between modes of accessing data from board, either locally (local keyword used) from any mounted device, or remotely (http keyword used). As usual with XML we can define our own definition, except those starting with "Streamer": Exclusively reserved for driving the packet injector.

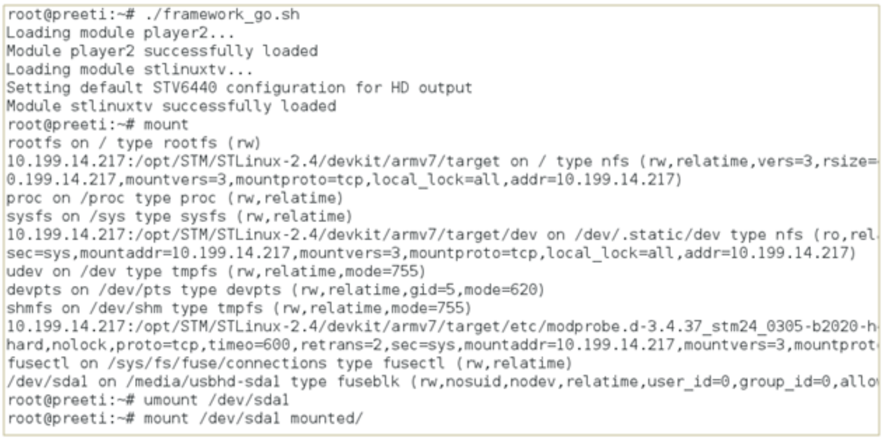
- A file selector. This is an XPath expression based on the contents of the Stream descriptions. It provides a list of files and applies the test case on them. The name of the python module containing user script (use-case-script node) and in the following launching order:
  - pre-command: System call such as an fbset
  - use-case-pre-launch: Python function accessed before program is launched (let prepare the context)
  - use-case-execution: Python function performing the test itself
  - use-case-post-launch: Python function after application exits, let perform real time verdicts with full statistics
  - post-command: System call
- All python function receive as parameter (optionally) the Python dictionary of the stream properties to know duration, resolution, codex, and so on They return a value that is passed of failed, ruled by the framework to the reporting.

4.5 How to Execute Test Cases Uses Test Framework:

This include some basic steps required to be followed in order to launch even a single test case using this framework. These are considered mandatory to be done after setting up the environment and to be executed in a serial manner.

- Step 1: Boot the Board
- Step 2: View the UART logs via serial-relay
  Wait till the log shows login message by username

- Step 3: Telnet the TARGETIP (IP of Board) as root
- Step 4: Load the modules like media player, HDMI Manager etc.
- Step 5: Mount the SCSI Disk in root folder inside opt directory(shown in figure 4.2).



```
root@preeti:~# ./framework_go.sh
Loading module player2...
Module player2 successfully loaded
Loading module stlinuxtv...
Setting default STV6440 configuration for HD output
Module stlinuxtv successfully loaded
root@preeti:~# mount
rootfs on / type rootfs (rw)
10.199.14.217:/opt/STM/STLinux-2.4/devkit/armv7/target on / type nfs (rw,relatime,vers=3,rsize=
0.199.14.217,mountvers=3,mountproto=tcp,local_lock=all,addr=10.199.14.217)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
10.199.14.217:/opt/STM/STLinux-2.4/devkit/armv7/target/dev on /dev/.static/dev type nfs (ro,rel
sec=sys,mountaddr=10.199.14.217,mountvers=3,mountproto=tcp,local_lock=all,addr=10.199.14.217)
udev on /dev type tmpfs (rw,relatime,mode=755)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620)
shmfs on /dev/shm type tmpfs (rw,relatime,mode=755)
10.199.14.217:/opt/STM/STLinux-2.4/devkit/armv7/target/etc/modprobe.d-3.4.37_stm24_0305-b2020-h
hard,nolock,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=10.199.14.217,mountvers=3,mountprot
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
/dev/sda1 on /media/usbhd-sda1 type fuseblk (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allo
root@preeti:~# umount /dev/sda1
root@preeti:~# mount /dev/sda1 mounted/
```

Figure 4.2: Mounting Hard Disk

- Step 6: Change the directory to mainline manifest's test framework directory.
- Step 7: Export some of the environmental variables in initial.sh file. These variables are:
  - SERVERIP : IP of host machine
  - TARGETIP : IP of the board
  - JEI : STMC (ST Micro Connect) IP
  - GWIP : Gateway IP
  - NFS-SERVER : Network File System IP
  - STREAM-SERVER : Used for special cases of HTTP
- Step 8: Source the above mentioned file

- Step 9: Execute the script "main.py" along with required arguments to start the tests.

  Even we can give some options along with above command. If we give option 'initialboot" then we have to follow from Step 6.

- Step 10: See the Reports in RESULT folder of that particular test framework.

4.6 By Test Framework:

There are some xml files used for executing each of the test cases. Each of the have different purpose.
Declarations are made of several xml files listed in the following paragraphs:

- Root
- Stream descriptions
- Stream locations
- Application abstraction
- Test calls
- Board configuration

4.7 Test Result:

The Test Framework provides a test status based on the launched application execution. These verdict are very important in order to send reports for a particular tag.

4.7.1 VERDICT DESCRIPTION

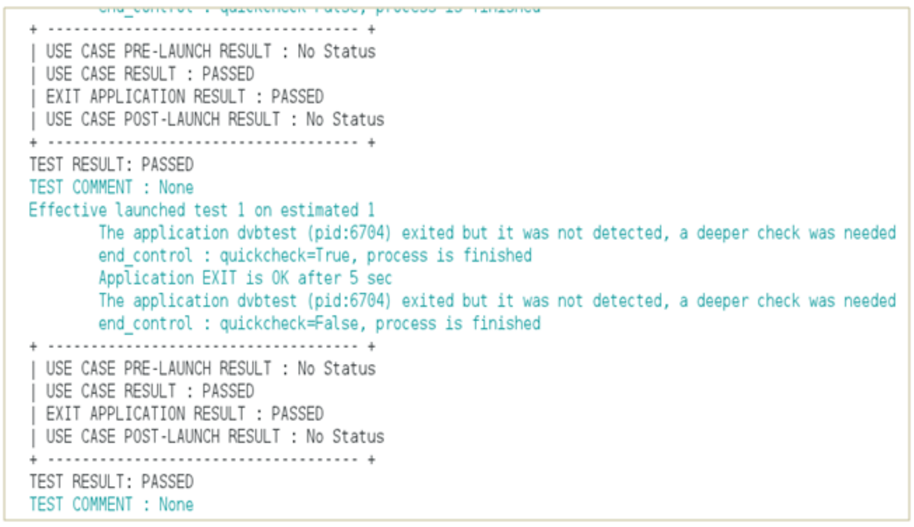
The verdict provided by test framework is important from point of view of providing cause of failure. As shown in figure 4.3 there are different types of verdicts provided by test framework.

- ABORTED: User termination of the test Execution
- CRASHED: Execution leading to a KERNEL crash, system need to reboot
- TIMEOUT: The application didn't succeed to exit, need to reboot
- XFAIL: The test result is fail but it is a known issue and specified in a file
- FAILED: Execution or verdict has failed but system is still considered as safe. Test script is not existing/defined
- SKIPPED: Test listed in the test plan but not executed
- PASSED: Execution and verdict (if exist) OK
- MISSING: The file is missing
- XPASS: The result was expected as False but it is passed
- Verdict by Test Framework

4.8 Types of Test suites Included In Test Framework

4.8.1 Sanity

This is the most basic test suite which is executed every time. For integrating a patch into the new tag it is required that the patch should pass the whole sanity procedure otherwise it won't be considered for review by the integrator.



```
        end_control : quickcheck=False, process is finished
+ ----------------------------------- +
| USE CASE PRE-LAUNCH RESULT : No Status
| USE CASE RESULT : PASSED
| EXIT APPLICATION RESULT : PASSED
| USE CASE POST-LAUNCH RESULT : No Status
+ ----------------------------------- +
TEST RESULT: PASSED
TEST COMMENT : None
Effective launched test 1 on estimated 1
        The application dvbtest (pid:6704) exited but it was not detected, a deeper check was needed
        end_control : quickcheck=True, process is finished
        Application EXIT is OK after 5 sec
        The application dvbtest (pid:6704) exited but it was not detected, a deeper check was needed
        end_control : quickcheck=False, process is finished
+ ----------------------------------- +
| USE CASE PRE-LAUNCH RESULT : No Status
| USE CASE RESULT : PASSED
| EXIT APPLICATION RESULT : PASSED
| USE CASE POST-LAUNCH RESULT : No Status
+ ----------------------------------- +
TEST RESULT: PASSED
TEST COMMENT : None
```

Figure 4.3: Test Framework Providing verdict

- This Test is used when there is any new tag arrival.
- It included all generic usecases which are required to be tested to check that earlier functionalities are not breaking with new development.
- It include all usecases related to user interface.
- It is executed using a makefile which provide target for running the sanity along with path to root files provided as argument.
  make path-to-root-file
- Results are generated as html file and can be obtained with all details through;
  make test-framework-results-summary

4.8.2 LSST : Light Sub System Test

After sanity another test suite which is executed most of the time is LSST. It include most basic types of use cases. It include use cases related to trickmode, mediaplayer, IP etc.

- It included many use cases related to:

  a. LIVE

  b. DISPLAY-PIP-LIVE

  c. DISPLAY-PIP-MIXED

  d. DISPLAY-ASPECT-RATIO

  e. DISPLAY-SUBTITLE

  f. DVR-TIMESHIFT

  g. DVR-TRICKMODE

  h. PLAYBACK-CONTAINERS

  i. PICTURES

  j. IP-VOD-HTTP

  k. IP-VOD-RTSP

  l. IP-LIVE

  m. IP-PIP

- It include around 74 test cases.

- For all type of use cases a particular result directory is generated which include:

  a. HTML file

  b. XML file

  Through xml file an html file is generated using :

  xsltproc path-to-session-file xml-file report.html

- A tar file is generated including all reports of all use cases.

  tar -cvf abc.tar file-containing-all reports

4.8.3 FSST : Full Sub System Test

When any code freeze arrives it is required to check all types of applications on full scale i.e. many use cases of all types of sources and various container streams are executed.

- It include all test cases of all types.
- It include around 600 and more usecases in around 23 root files.
- It is required to be done when there is a need to test a full release before delievery.
- It include all test cases related to :
  a. Media
  b. DVR
  c. User-Ops

4.8.4 LSST for "INFRA" kernel test cases

This is very small test suite. Any patch related to kernel nedd to pass this test procedure before integration.

- For running LSST for kernel a special procedure is followed:
  - first of all infra module is built.
    make infrastructure
  - Then module is installed.
    make .modules-install-infrastructure
  - Environment of infra test is set up using:
    make .install-infrastructure-tests
  - Then Few files explictly used for infra are copied at root.
  - Board is booted and telnet is done to get connected to board.
  - Copied files are sourced and by this LSST get started.
  - Without any display all test cases get executed.

4.8.5 MAT : Mainline Acceptance Test

This is executed using make file. Target included in make file starts the tests by taking prerequisites as root file and calls file. Python command is also included in make file in order to start the test.

- Before giving any tag for any SDK there is a need to run a basic suite which is MAT.
- If the verdict provided by MAT is FAIL the that particular tag cannot be released.
- It include test cases related to:
  a. STLINUX
  b. STMF

4.9 COVERITY

When source code is huge and include lots of variables, conditions on it for different purposes the it is must to check that all variables are initialized or either they are used or not. For that coverity tool is used.

- It is a Tool used to check :
  a. Any variable in source code is undefined.
  b. Any variable is not used but it is declared. c. Any variable is not correctly defined.

4.9.1 Steps to RUN COVERITY

- Clone the Coverity from codex page to your Linux Machine (we support both 32 bit and 64 bit machines.
  git clone ***.git

- Coverity gather all it's data while compiler compiles your code. Therefore,Make sure you are able to compile SDK2 before Coverity should be used. We will use standard SDK2 build procedure while using Coverity.
- Before running Coverity, make sure your have run 'make clean' type on your code.This will ensure when we run Coverity, code is actually getting compiled and Coverity can gather data for analysis
- On the console where you are buuilding your sdk2 code, set following env variables:
  For 32 bit machine:
  export COVERITY-ROOT=/your/coverity/path/upto/cov-analysis-linux-6.6.2
  For 64 bit machine:
  export COVERITY-ROOT=/your/coverity/path/upto/cov-analysis-linux64-6.6.2

  export PATH=$COVERITY-ROOT/../scripts$:COVERITY-ROOT/bin:$PATH$
  $exportPATH = /opt/STM/STLinux - 2.4/devkit/armv7/bin/$:PATH
- Coverity is ready to be used now! On sdk2, all modules can be built using 'make module'. In order to use Coverity, use command 'coverity-build-linux.sh make modules' This command will make sure Coverity captures all it's data and then starts analysis on gather data.

4.10 Innovation

4.10.1 Introduced HTTP Support

In case of streaming through any host, it is done in two ways:
a. By Dektec Card
b. By cvlc

- By Dektec Card

- By CVLC:
  - In case of cvlc command connection with host is made via cvlc command. In that scenerio a cvlc command is launched on host, it get connected via socket address to respective host then there is need to just launch the application using http. Earlier all this work for http was done manually only. As it is a http source then it must be run on host, because of this application command contain host IP.
  - Problem Statement:
    - In Test Framework support was provided for protocols like UDP and RTP but there was no support provided for http in LSST (Light Sub System Test).
    - No support was there for generation of http command via cvlc
    - Init functions defined for every protocol in LAUNCH FILES was also not there for http. Because of that it was not possible to launch http command via application in Test Framework.
  - Solution adopted:
    - Cvlc command support was provided with the help of command generated via vlc for http that command is included in python file to provide http support via Test Framework.
    - To generate command to be launched on target via application "INIT" function is included in launch files to enable streaming using http protocol via cvlc in Test Framework.

4.10.2 Added New Usecases In Test Framework

- Problem Statement:
  - Support for USECASES related to pause state were missing.

- How a pipeline behaves in pause state when different User Interface commands are provided as input.
- State of pipeline need not to be changed for all types of user interfaces

- Solution Provided: Introduced new Usecases by which pipeline behaviour can be observed in PAUSE state.
  - PAUSE-FAST-TRICKMODE
  - PAUSE-FAST-TRICKMODE-RESUME
  - PAUSE-SEEK-RESUME
  - PAUSE-REVERSE-TRICKMODE
- All usecases are included in:
  - Root files
  - Call files
- New function for pause state is introduced in python file which parses that xml files.

4.10.3 Solved bugs related to Test Framework

- Solved issues related to wrong names :
  - Problem Statement: Descriptions, calls and location files of test framework were having wrong file names of streams (mainly of media player streams) due to which none of the test case was running.
  - Solution Provided: To solve these all containers streams names are modified according to the name of streams available on server.
- Solved issues related to stream duration of 60 secs.

- Problem Statement: In FSST there was no provision provided for handling streams of duration less than 60 seconds. Due to which usecases running on such streams were breaking.
- Solution : A support for stream duration less than 60 seconds is provided in functions and modified according to use cases utilizing those streams.

• Introduced LOCAL-SERVER-LOCATION=IP of server for streaming through local server.

  - Problem Statement: In http usecases due to server issues buffering problem was observed. As a result all such use cases verdict provided by Test Framework were coming FAILED.
  - Solution: A new server location is needed to solve this problem. All streams are uploaded on new local server and a new VARIABLE is used to provide its support, that variable is LOCAL-SERVER-LOCATION. This variable is used in location file and exported in parameter file. As a result of this if any one wants to change the server ip from which stream is to be fetched then he has to export new ip from parameter file.

• Solved issue of RAP INDEX issue by providing no-remove-id option as an argument in xml file.

  - Problem Statement: In DVR usecases of RAP were breaking because there was no need of providing pids as it reads PIDS from ifo file.
  - Solution: A new additional parameter is added in root file called as remove-id-option by which mo pids will be included in the command that will run on target.

4.11 Summary

This chapter include prerequisites required to start the test framework. It also include different types of test suites used for doing regression testing. Along with that need of coverity tool is also covered. Innovation work include solved bugs of test framework and included new usecases in test framework.

Chapter 5

Use Cases Implemented in Digital Video Recording

5.1 Need of DVR

Recording a particular program is now a days normal for any user but why it is required is to be understood. DVR's are better than VCR's for these primary reasons [1]:

- No more tapes to change every day
- Better picture quality on playback.
- Instantly search through recorded images by time and date, without spending hours in front of a VCR trying to find the event you are looking for.
- Extended storage capabilities without long-term maintenance.

5.2 Digital Video Recording:

This is done by various ways. Even it include a special element called recorder is included in pipeline when DVR usecases are executed.

- Use Cases in DVR:

  a. Recording

  b. Playback

  c. Time Shift

  d. Live Playback

  e. Live Pause

  f. Resume

  g. Catch Live

  h. Trick Modes

  i. Indexing

  j. Review Buffer and Security

- Implementation in Gstreamer :

  a. Live Viewing

  b. Recording and Playback

    - TS File
    - IFO file
    - Circular File

  c. Multifile handling

  d. Trickmodes and Seek

  e. Time Indexing

  f. RAP Indexing

  g. Timeshift

  Live Pause Resume Catch Live

5.2.1 Recording

This is the most easy usecase. As shown in figure 5.1, in this case read and write pointer move together and because of that we are able to watch live along with recording going on.

- Also known as background recording if no playback ongoing or settop box in sleep mode
- Record any subset of PID from a single MPTS
- Source: SAT/CAB/TER or IP
- To local storage : HDD, Sata, USB,
- Different record option:
  Scheduled record Start record from live and onwards Start record from past (beginning of the movie) using the review buffer
- Option of playing same channel while recording (Watch and Record)
- During the record application may grab sample image to build custom preview

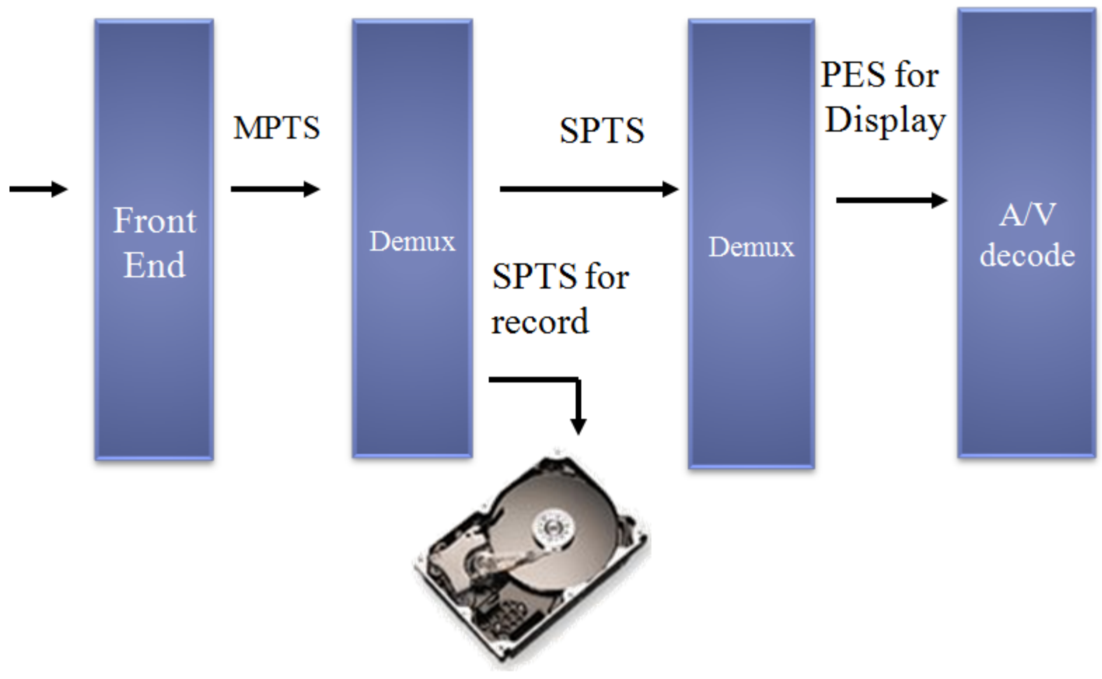


Figure 5.1: Recording in DVR

5.2.2 Playback

From local storage: recorded completed. Playback possible while record is still on going (equivalent to time shift use case, shown in figure 5.2)

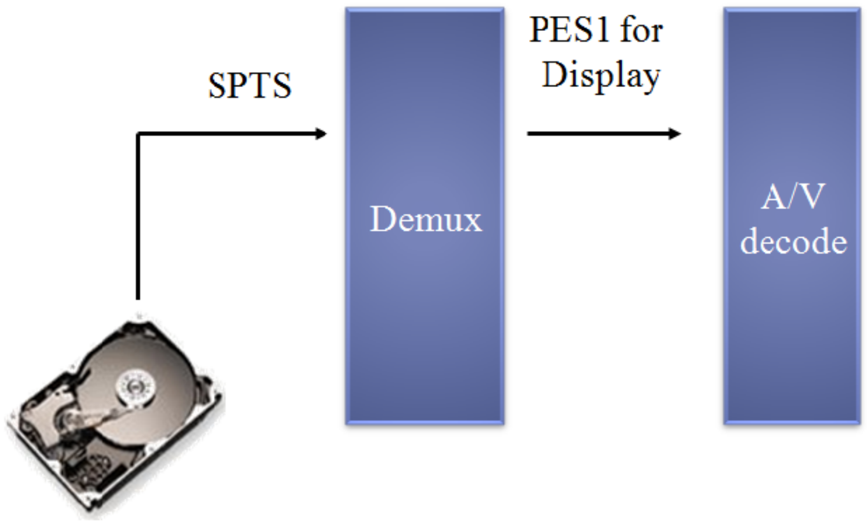


Figure 5.2: Playback in DVR

5.2.3 Live Playback in Timeshift

Live Playback : In this case Read pointer comes in line with Write pointer. Because of this instead of reading from Hard Disk directly data is going live from source and recording get stoppped (shown in figure 5.3).

5.2.4 Pause in Timeshift

Pause Live: Possible to pause the live. As shown in figure 5.4, in this case live data get stored in Hard Disk in a file and Write pointer continuously get incremented. In that scenerio we see same frame on screen.

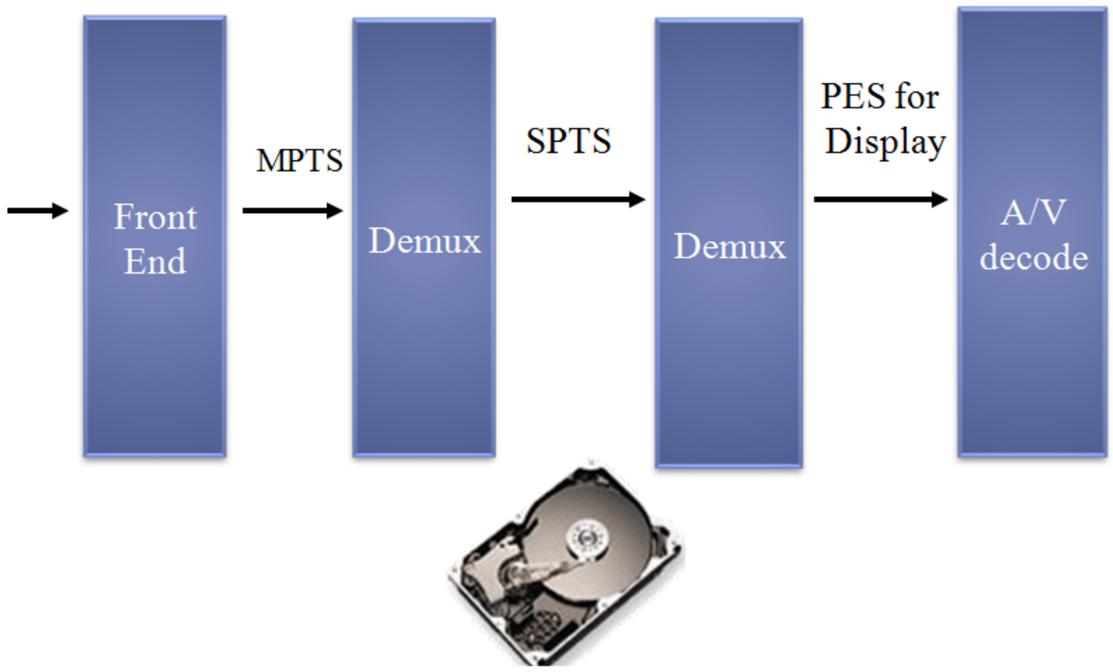


Figure 5.3: Live Playback in DVR

5.2.5 Resume in Timeshift

In this Case we start reading data from Hard Disk. Along with Write pointer Read pointer start increasing and till the point when Read pointer is behing Write pointer everything will be read from disk. Point when Write approaches Read catch live will be there, shown in figure 5.5

- Resume
  - Resume playback from the recorded file
  - Trickmode possible with in the recorded stream (limited to the duration of the storage on disk)

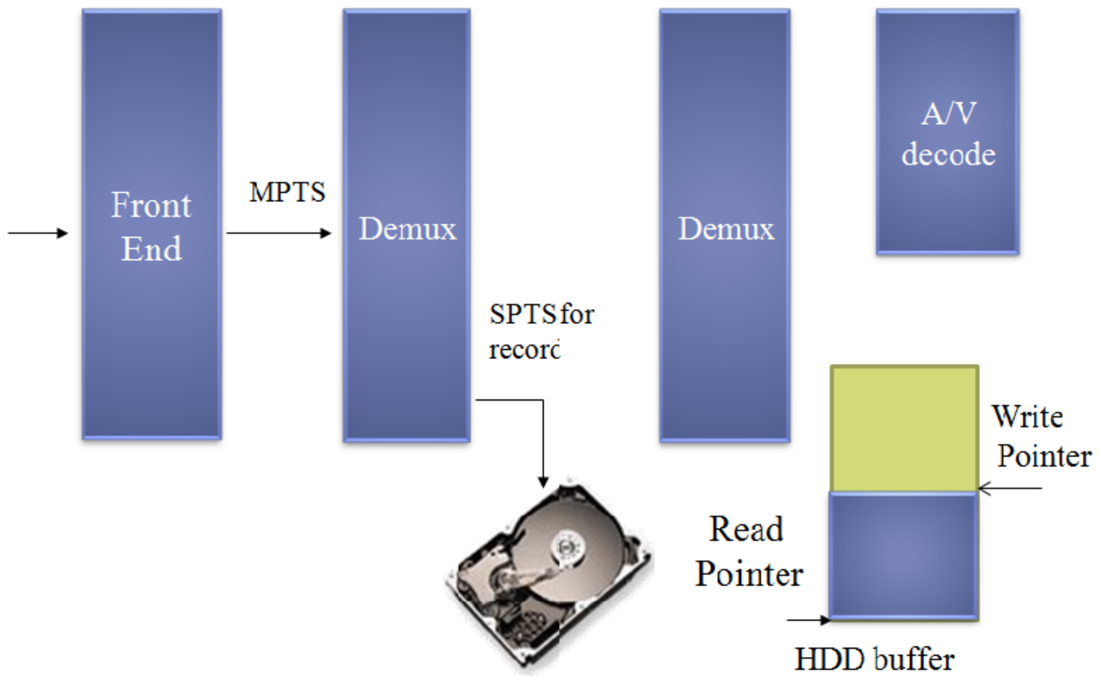


Figure 5.4: Pause in DVR

5.2.6 Catch Live in Timeshift

Externally also catch live is possible which is done by '@' symbol. So, these are the basic requirement when nature is to be observed in pause state because before this read pointer must be running before write pointer and on doing this write pointer will be ahead of read pointer(shown in figure 5.6).

- Catch Live
  - Back to live (in fact minimal achievable latency)
  - Smooth transition: no artifact on display
  - Record remains active (possible to go backward again)

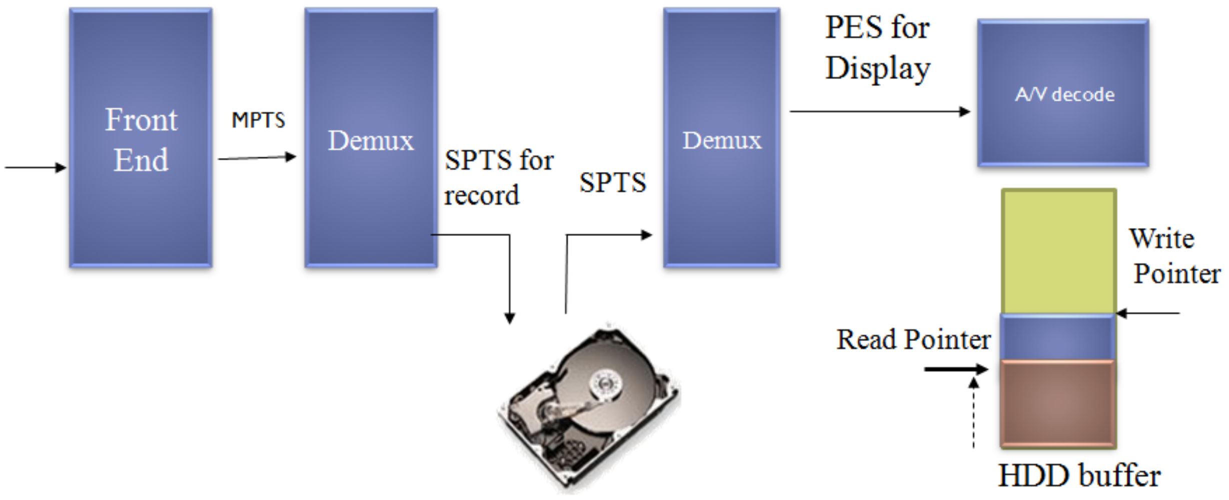


Figure 5.5: Resume in DVR

5.3 Innovations: Need to check different usecases

5.3.1 Problem Statement

- On arrival of new patch made by use of DIFF utility there is a need to do regression testing by ;

  a. Manually

  b. Testframework

  For that, one should be aware of all type of use cases that can be tested.

5.3.2 Solution:

- Applied Usecases related to CHANNEL ZAPPING while recording is going on: In this solution one channel is playing and recording started, along with that user operation is provided for channel zapping via 'u' and 'd'. Due to this while recording channel zapping is supported or not, avsync is maintened or not is observed.

- Applied Usecases related to DISPLAY ON while recording is going on:

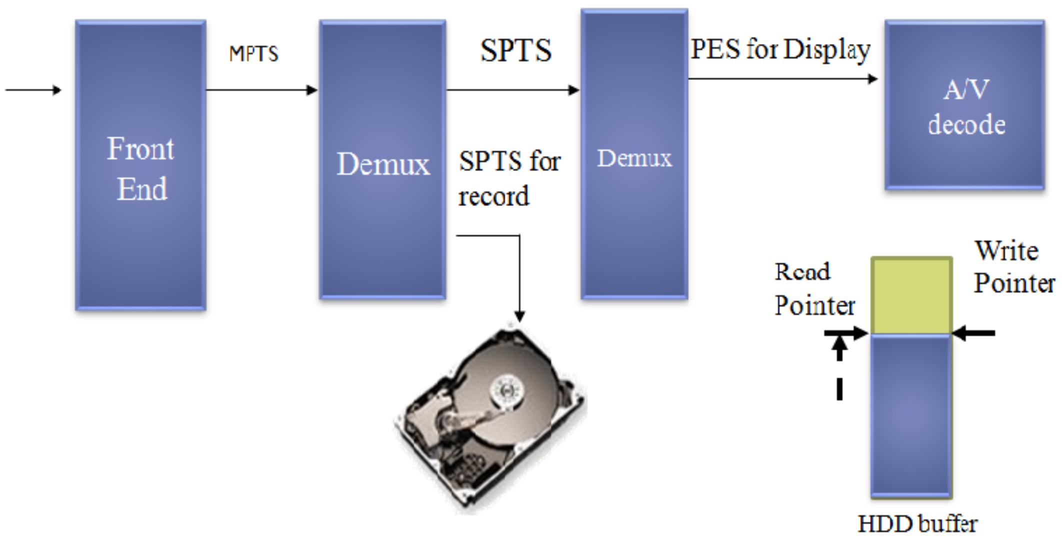


Figure 5.6: Catch Live

When recording is started then with user operation '*' display can be off and on. It can be observed that when recording is going on and we switch off a device then what will be the effect.

- Tested New and many uses of CATCH-LIVE:
  - Catch live by trickmode i.e. by fast forwarding read pointer try to overtake write pointer. In that case catch live will be observed.
  - Direct catch live: when timeshift is going on and playback started the catch live by '@' is another usecase.
  - Catch live by seek beyond the range: When try to jump beyond the range present in rap file will give catch live.

5.4 Summary

This chapter include introduction to DVR, its different usecases and how the work is further exceeded in order to check the patch and find out any bug related to DVR.

It also include work done related to bugs in DVR as well as different ways adopted to run use cases of DVR. Innovation work include testing of all patches of DVR and searched new usecases, as well as found new issues in catch live in case of DVR.

Chapter 6

Implementation of Circular Buffer in DVR

6.1 Requirements

Recording in a file of a user configurable size, called Circular File Size. Recording will be linear till, the Circular File Size is reached. After Circular File Size is reached, the data in would be overwritten from the beginning of the file, on consecutive writes. Read pointer , for playback would be modified/advanced so as not to show the overwritten data. Regular update of metadata in IFO.

6.2 Current Design

Current Design has incomplete implementation of circular buffer. No overwrite takes place, when Circular File Size is reached, we drop data. Circular wrap of read and write pointers over the Circular File Size is checked before every write to ts file, and we may either write the ts or drop data. IFO is updated only once when the recording is closed.

6.3 New design

6.3.1 Metadata Update

- New fields added to the IFO
  - Circular File Size(CircularFileSize): User configurable size of file
  - Circular Start Position(CircularStartPosition): Read offset for playback
- Advantages:
  - IFO Update is to be synchronized with writes to the TS file, so we update IFO after every write to the disk , in the recorder callback.
  - In case of accidental power failure, file would be playable since metadata of alteast last 100ms is available
- Disadvantages
  - Disadvantage in making a new task for ifo update.
  - This will burden the system, with a lot of write system calls at regular intervals. This may have adverse effect on the life an performance of storage disk over a long time

6.3.2 Recording

As shown in figure 6.1, how circular buffer is implemented in recording usecases, and how pointers move as buffer fills is shown.

- Recording to be done in a fixed space(user specified file size) on the disk
- Attributes to be manipulated
  - Read pointer: for Playback
  - Write pointer: updated after every write to the disk.

  - Circular file Size: maximum size of file permitted in recording, data must be overwritten from the front in order to maintain continuity in recording.
  - Circular Read Start: Valid reading offset, when circular overwrite has occurred

- When the circular File Size is reached, circularize, the write pointer and overwrite the data from the beginning.
- Update Circular Read Start after every write, following the circular overwrite
- Update ifo after every write

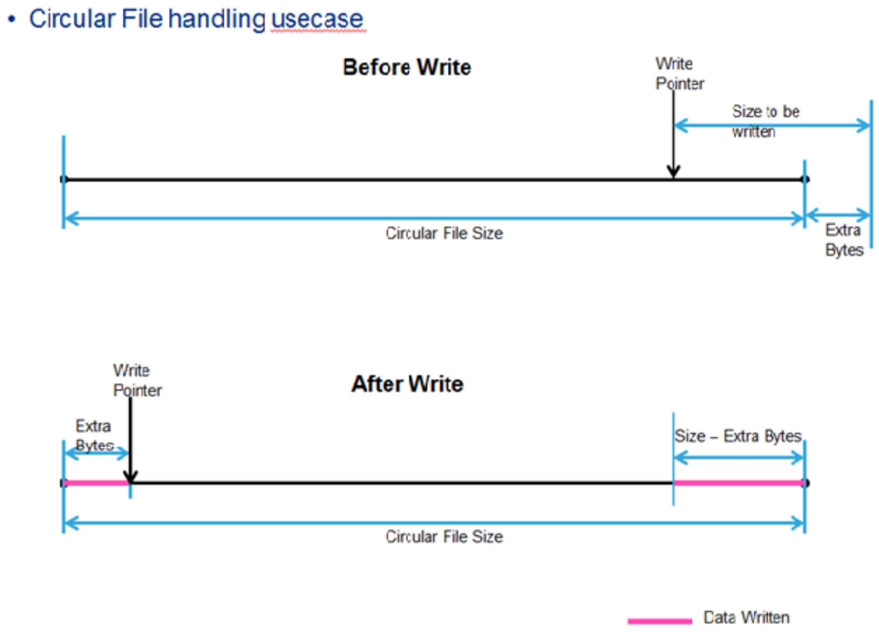


Figure 6.1: Recording Implementation using Circular Buffer

6.3.3 Timeshift

This usecase may have both recording as well as playback. Recording and ifo update is also done in this usecase. Circular read start will be read from the ifo, and will update the read pointer. Write pointer is updated linearly upto Circular File Size, and after the wrap, it is circularised to overwrite the data from the beginning. IFO updated after every write. Every consecutive write after circular wrap will,update Circular Read Start.If write close to read, while playing, raise event to application and stop recording(shown in figure 6.2).

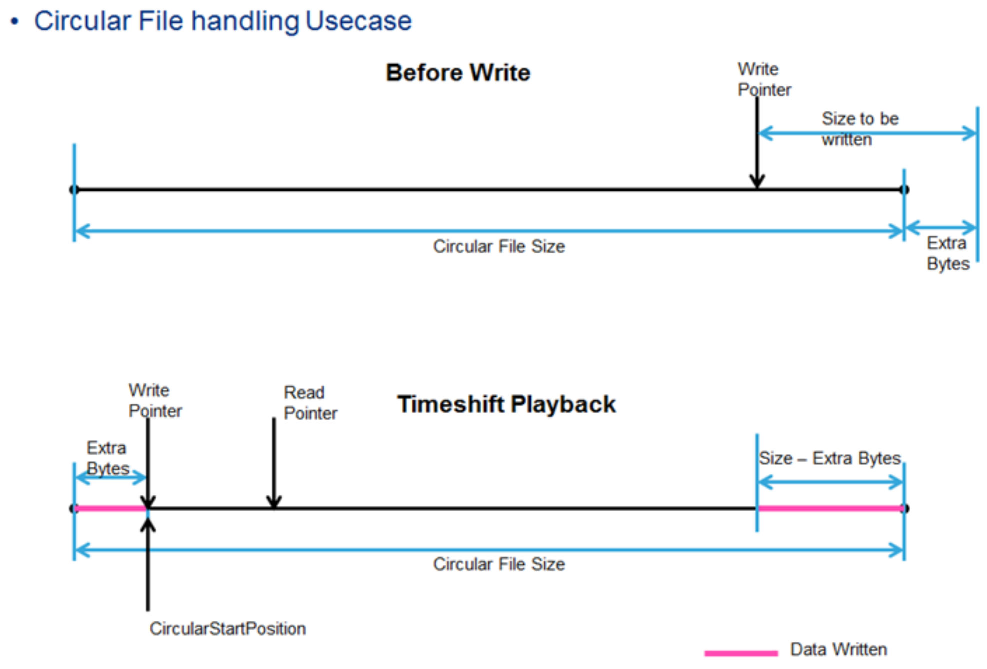


Figure 6.2: TimeShift Implementation using Circular Buffer

6.3.4 Playback

In playback, IFO is to be given priority over PAT/PMT. Overwrite in circular file, would have overwritten PAT/PMT written at the starting of linear file. We update read pointer , with Circular Read Start read from the ifo file, as shown in figure 6.3.

6.3.5 Timeshift Play Issues

- When playback in timeshift is slower than the rate at which data is written to file. Write pointer approaches read pointer after wrap over circular file size.

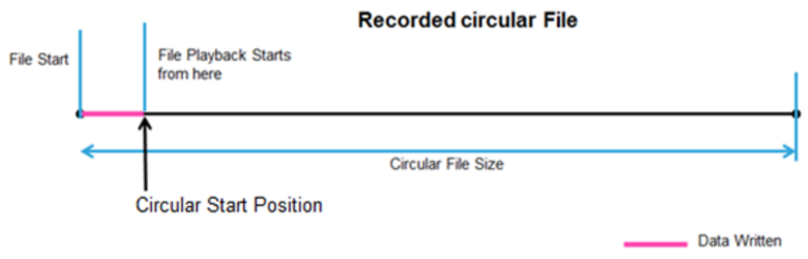


Figure 6.3: Playback in Circular Buffer

E.g.. Slow forward during timeshift.

- In the usecase , of slow forward timeshift(less than 1x) , the write pointer will gradually start approaching read pointer since reads are slower and writes are faster. This will give rise to a condition where write is close to read and may tend to overwrite it.

- Other problem arises in the case of slow or fast backward trick mode(shown in figure 6.4). In this case the read pointer moves backward and write pointer moves forward, and hence they may converge to the situation of write close to read

- In this usecase, when write is faster than the read (play at 1x) due to bitrate fluctuations

6.3.6 Timeshift Pause Issues

- In this usecase , we pause the recording and the wait for a long time, till the wrap over circular file occurs and write pointer comes near to read pointer.

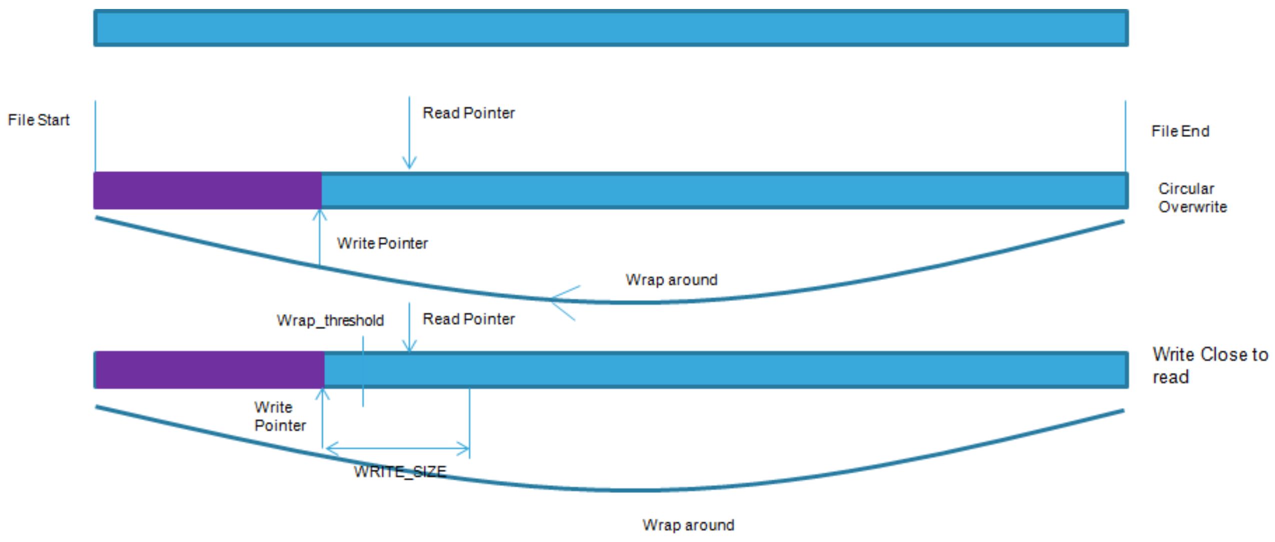


Figure 6.4: Timeshift Play Issues in Circular Buffer

6.4 Work Done: Circular Buffer

6.4.1 Usecase 1: Related to BOS in Reverse Trickmode

It is required to test the usecase of reverse trickmode in circular buffer carefully because when file get overwritten in that scenerio what should be the behaviour.

- Linear : In case of normal file when reverse trickmode is applied the at the starting of file beginning of file is received.
- Circular : When file become circular in that case when reverse trickmode is applied in timeshift then instead of getting BOS i.e. Beginning of Stream Catch live is observed.

6.4.2 Usecase 2: Behaviour on SEEK to '0'

In SEEK to '0' actually a command is send to application in which its position is send where to jump. In ever type of SEEK same scenerio is there. A position is send accordimg to seek query and a position is sent related to seek position with help of

RAP or TIME INDEX file.

- Linear : In case of normal file on SEEK to '0', stream start playing from the start of file from where recording started.

- Circular : In case of Circular file, as file is circular so when buffer is filled first time then overwriting started. Now, when we do SEEK to '0' then instead of starting from start of file it jump to a position from where data is not overwritten i.e. it jump from offset.

6.4.3 Usecase 3: RAP file Entry

In case of DVR, two files are used for TRICKMODE and SEEK in order to obtain relative position. These are:
1. IFO
2. RAP
There are entries which are used to find the position.

- Linear : In normal files, RAP i.e. Random Access Point, these are points when we receive I-FRAMES in GOP(Group of Pictures) , so all entries are checked and then position is decided.

- Circular : In circular files in some conditions after 60 sec recording RAP entries were missing from RAP files, so Time index files is used to get the relative position.

6.5 Innovation : Playing a Circular file through vlc

6.5.1 Converting a Circular File into a Linear File

For any file to play there is need of PAT(Programme Association Table) and PMT(Programme Map Table) which the respective pids required to play a file.

- In case of any linear file starting 1024 bytes contain PAT/PMT table. But when there is a case of circular file, in that case due to overwriting starting 1024 bytes which contain the necessay information get lost and as a result of this it is not possible to play this file.

- There are two methods to play this file:
  1. Through IFO file
  2. Converting circular file to linear and then adding PAT/PMT..

- An ifo file contain the necessary PID information. In case of DVR it is generated and used for playing a recorded file. With the help of this file PIDs will be obtained to play the file.

- Problem Statement:

  – When file become circular that is a portion of file get overwritten in that case all header information get lost which is required to run a file i.e. all PIDS required to play are lost

- Solution Provided : Circular to Linear transformation

  – With the help of shell and C a tool is designed which convert a circular file to linear file.

  – Now there is need of PAT/PMT to be added. A recording of same channel is done and starting 1024 bytes is seperatly stored in a file.

- This file is added before that linear file which is obtained by converting circular file to linear.
- Basically a tool was needed to be designed to extract
  1. Extract PAT/PMT
  2. Obtain Overwritten portion of circular file in a seperate file
  3. Convert Circular file to linear
  4. Add PAT/PMT to linear file

6.6 Summary

This chapter includes the basic concepts of circular buffer. Circular buffer led to creation of circular file which is some what different from linear file because its header part is overwritten when buffer get filled. So, to retrive any sort of information for trickmode some addition files are required. Even it is not possible to play circular file with vlc because header information is missing in it. A tool designed to convert circular file to linear which makes circular file possible to play through vlc. Innovation work include tool made for circular buffer to convert circular file into linear, solved bugs related to circular buffer and testing of circular buffer patch.

Chapter 7

Application Development By : Gstreamer

7.1 What is gstreamer

Gstreamer is a multimedia framework. As shown in Fig7.1 a multimedia framework is a software which provides an abstraction layer for the developers of multimedia applications.GStreamer is a library applications just link to it to get functionality.
To build a multimedia application, developers need some basic components. The components include sources and destinations of media stream as well as some manipulators. These components in the terminology of gstreamer are called Element.
An element can have several roles. An element which is the source of stream is called source element, a manipulator element is called filter element, and finally the destination of stream is called sink element. Having a complete set of elements, one can implement any multimedia application.
Since multimedia framework must be extensible by users, gstreamer is implemented with a plugin- based architecture. In the language of gstreamer, a user can implement a filter which suits his application and uses it through gstreamer framework. In the following picture you can see the role of gstreamer.

- Provide an Object Oriented API to build multimedia systems
  - API for Applications
  - API for Plugins
- Provide policy/architecture/environment
  - Synchronisation
  - Processing States
  - Dataflow
  - Events, messages.
- Process large amounts of data quickly
  - Allow fully multithreaded pipelines
  - Deal with multiple formats
    - . Audio, video, text, bytes
    - . Anything else
  - Abstract hardware devices
- Handle live sources, clock slaving
  - Support for different clock providers
  - Support for non-linear editing
  - Different scheduling modes
  - Dynamic pipeline changes
  - Extensible by plugins and application
  - Events, data formats, queries, ...

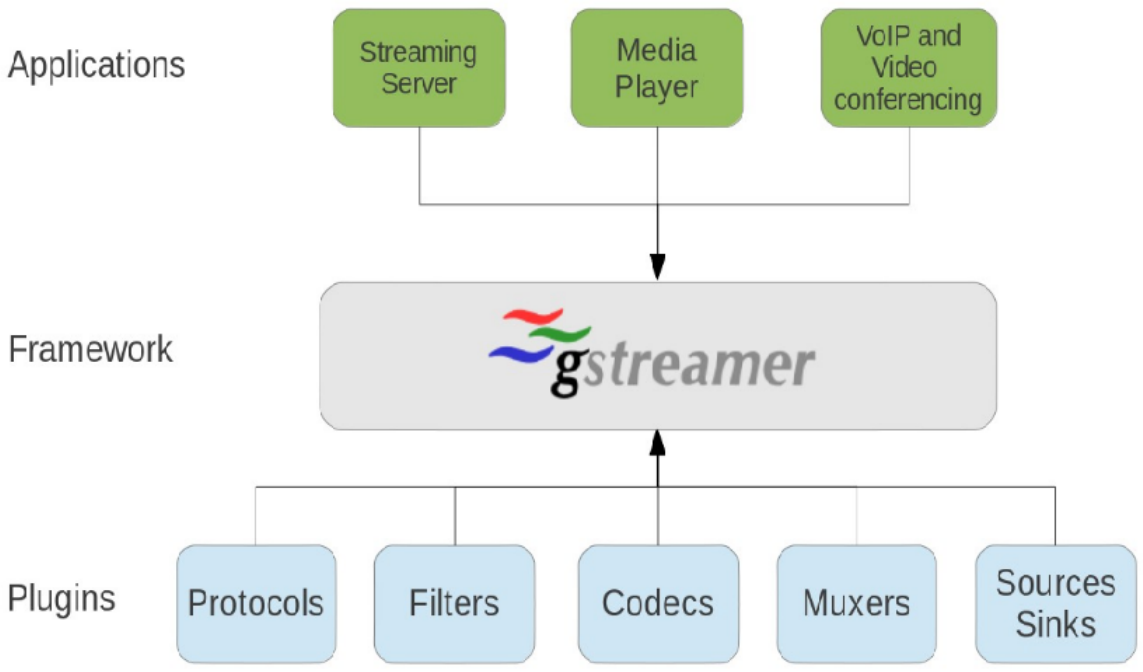


Figure 7.1: Gstreamer Framework

7.2 Basic Parts of Gstreamer

7.2.1 Elements

Elements are basic units of functionality. These are the main part of gstreamer which actually handle data and along with that provide facility of handling events (as shown in fig 7.3).

7.2.2 Bins

As shown in fig 7.4 bins are elements that contain other elements and allow multiple elements to be treated as one entity.

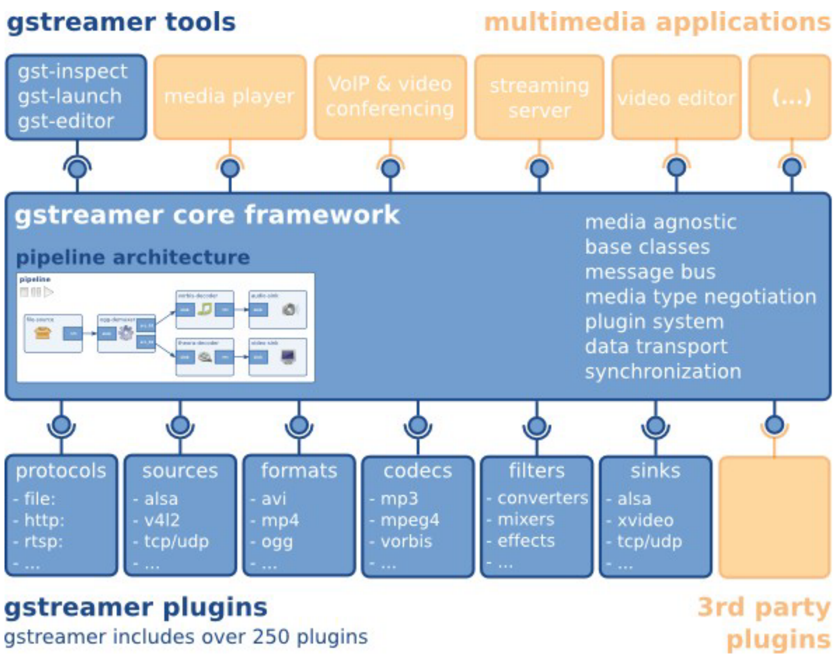


Figure 7.2: Overview of gstreamer

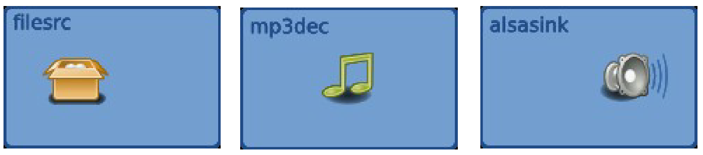


Figure 7.3: Elements in gstreamer

7.2.3 Pads

Pads create connection points between elements. Source pads produce data and Sink pads consume data(as shown in fig 7.5).

7.2.4 Pipeline

Pipeline is toplevel bin.Select and manage clock and running time (time spent in PLAYING). It also manage latency.Provides GstBus to application(as shown in fig 7.6).

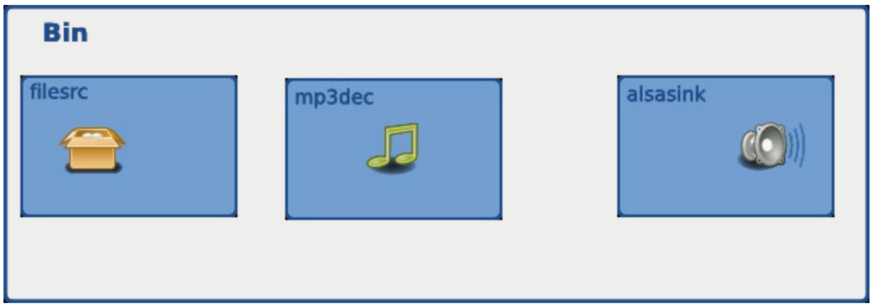


Figure 7.4: Bins in gstreamer

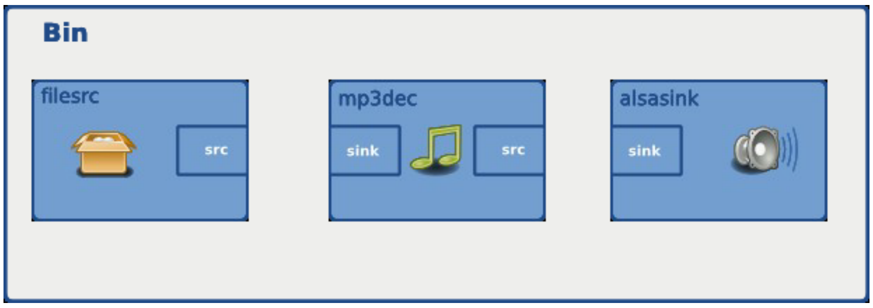


Figure 7.5: Pads in gstreamer

7.2.5 Bus

- Receives messages from elements
  - End-Of-Stream, error, warning, tags(title, track, etc.), state changes...
- Marshals the messages to the main thread
- Main method for the plugins to communicate with the appication
  - Signals (and property changes) are another alternative but threading is hard.

7.2.6 Controlling Data Flow

There are 4 states in which a pipeline can be:

- NULL
  - All resources deallocated, devices released, initial state.
- READY
  - Devices opened
- PAUSED
  - Buffers begin to flow, blocked in the sinks (clock is 'paused')
- PLAYING
  - All systems flow, clock is running
- Elements always go through intermediate states. Bins change state of children from sink to source. Pipeline selects clock and manages timing when going to PLAYING Elements can choose to start a thread (GstTask) to start dataflow.
- All we have to do is tell it to go to PLAYING, so something is happening behind the scenes.The data flow will happen in a separate thread. Thus states change asynchronously GStreamer provides thread-safety when using API functions on all of its objects.

7.3 Tools Introduced by Gstreamer

7.3.1 gst-launch

gst-launch is a tool that builds and runs basic GStreamer pipelines. In simple form, a PIPELINE-DESCRIPTION is a list of elements separated by exclamation marks (!). Properties may be appended to elements, in the form property=value.

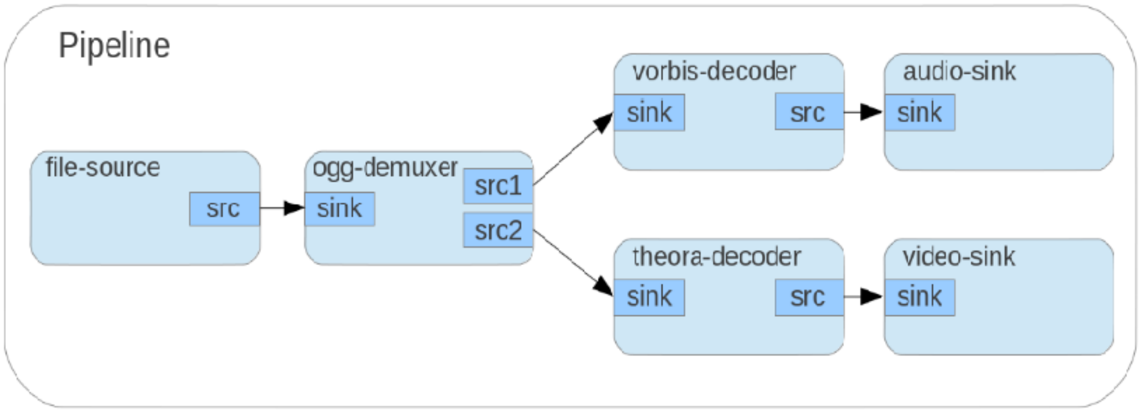


Figure 7.6: Pipeline

gst-launch is primarily a debugging tool for developers and users. We should not build applications on top of it. For applications, use the gst parse launch. function of the GStreamer API as an easy way to construct pipelines from pipeline descriptions.

- It is a debugging tool.
- It is used to prototype the pipeline.
- It shows the whole pipeline connection including
- gst-launch accepts the following options:
  - help: Print help synopsis and available FLAGS
  - -v, –verbose: Output status information and property notifications
  - -q, –quiet: Do not print any progress information
  - -m, –messages: Output messages posted on the pipeline's bus
  - -t, –tags: Output tags (also known as metadata)
  - -o FILE, –output=FILE: Save XML representation of pipeline to FILE and exit
  - -f, –no fault: Do not install a fault handler

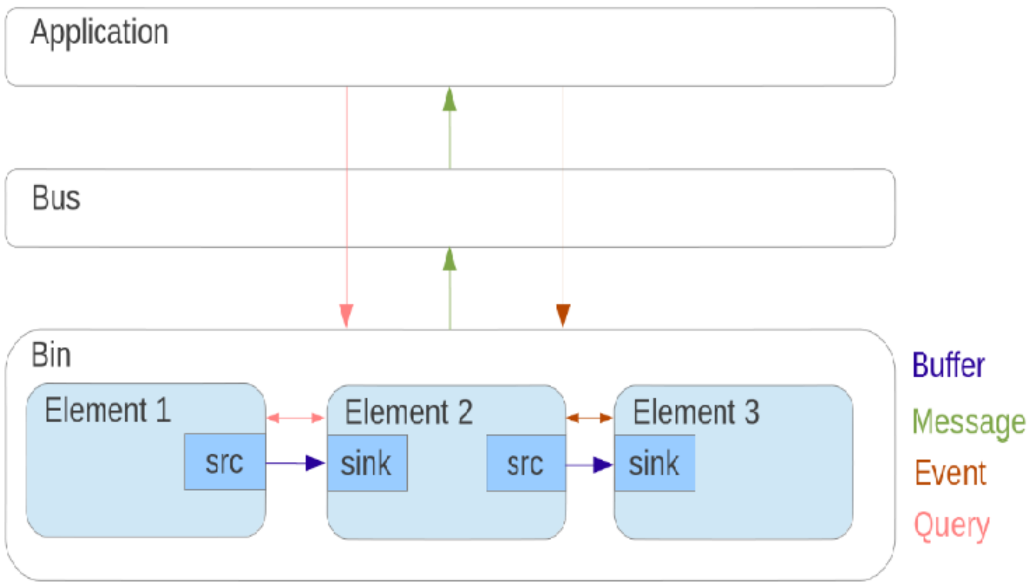


Figure 7.7: Pipeline

- -T, –trace: Print memory allocation traces. The feature must be enabled at compile time to work.

- Gstreamer Options: gst-launch also accepts the following options that are common to all GStreamer applications:
  - gst-version: Prints the version string of the GStreamer core library.
  - gst-fatal-warnings: Causes GStreamer to abort if a warning message occurs. This is equivalent to setting the environment variable G DEBUG to fatal warnings (see the section environment variables below for further information).
  - gst-debug-level=LEVEL: Sets the threshold for printing debugging messages. A higher level will print more messages. The useful range is 0-5, with the default being 0.
  - gst-debug-no-color: GStreamer normally prints debugging messages so that the messages are color-coded when printed to a terminal that handles

ANSI escape sequences. Using this option causes GStreamer to print messages without color. Setting the GST DEBUG NO COLOR environment variable will achieve the same thing.

- gst-debug-disable: Disables debugging.
- gst-debug-help: Prints a list of available debug categories and their default debugging level.

7.3.2 gst-inspect

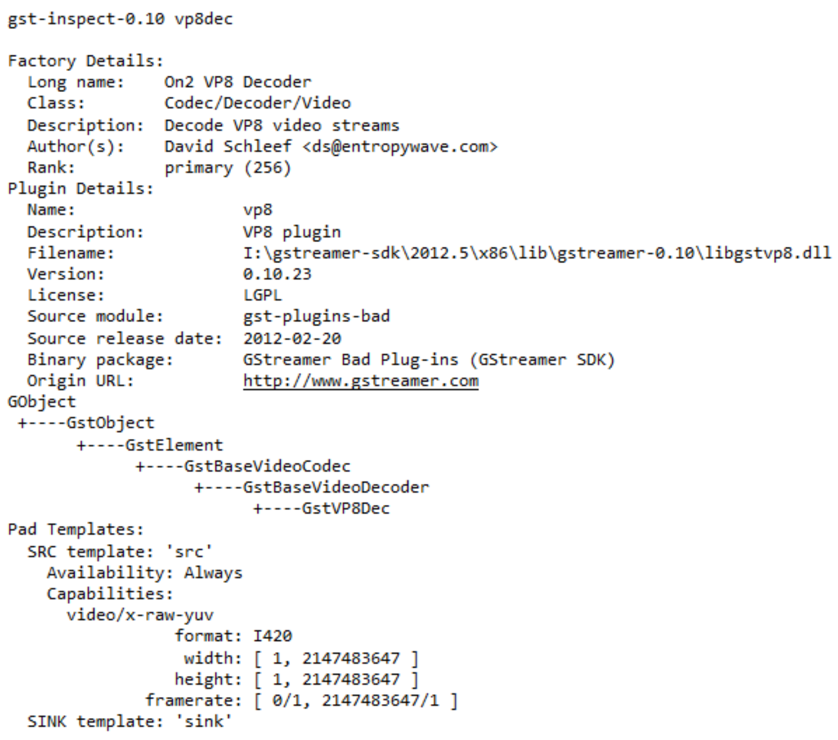
This tool has three modes of operation:

- Without arguments, it lists all available elements types, this is, the types you can use to instantiate new elements (as shown in fig 7.8).
- With a file name as an argument, it treats the file as a GStreamer plugin, tries to open it, and lists all the elements described inside.
- With a GStreamer element name as an argument, it lists all information regarding that element.

7.3.3 gst-discoverer

This tool is a wrapper around the GstDiscoverer. Media information gathering. As shown in fig 7.9, it accepts a URI from the command line and prints all information regarding the media that GStreamer can extract. It is useful to find out what container and codecs have been used to produce the media, and therefore what elements you need to put in a pipeline to play it.

Use gst-discoverer –help to obtain the list of available options, which basically control the amount of verbosity of the output.



```
gst-inspect-0.10 vp8dec

Factory Details:
  Long name:    On2 VP8 Decoder
  Class:        Codec/Decoder/Video
  Description:  Decode VP8 video streams
  Author(s):    David Schleef <ds@entropywave.com>
  Rank:         primary (256)
Plugin Details:
  Name:                 vp8
  Description:          VP8 plugin
  Filename:             I:\gstreamer-sdk\2012.5\x86\lib\gstreamer-0.10\libgstvp8.dll
  Version:              0.10.23
  License:              LGPL
  Source module:        gst-plugins-bad
  Source release date:  2012-02-20
  Binary package:       GStreamer Bad Plug-ins (GStreamer SDK)
  Origin URL:           http://www.gstreamer.com
GObject
 +----GstObject
       +----GstElement
             +----GstBaseVideoCodec
                   +----GstBaseVideoDecoder
                         +----GstVP8Dec
Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      video/x-raw-yuv
                 format: I420
                  width: [ 1, 2147483647 ]
                 height: [ 1, 2147483647 ]
              framerate: [ 0/1, 2147483647/1 ]
  SINK template: 'sink'
```

Figure 7.8: gst-inspect of Gstreamer

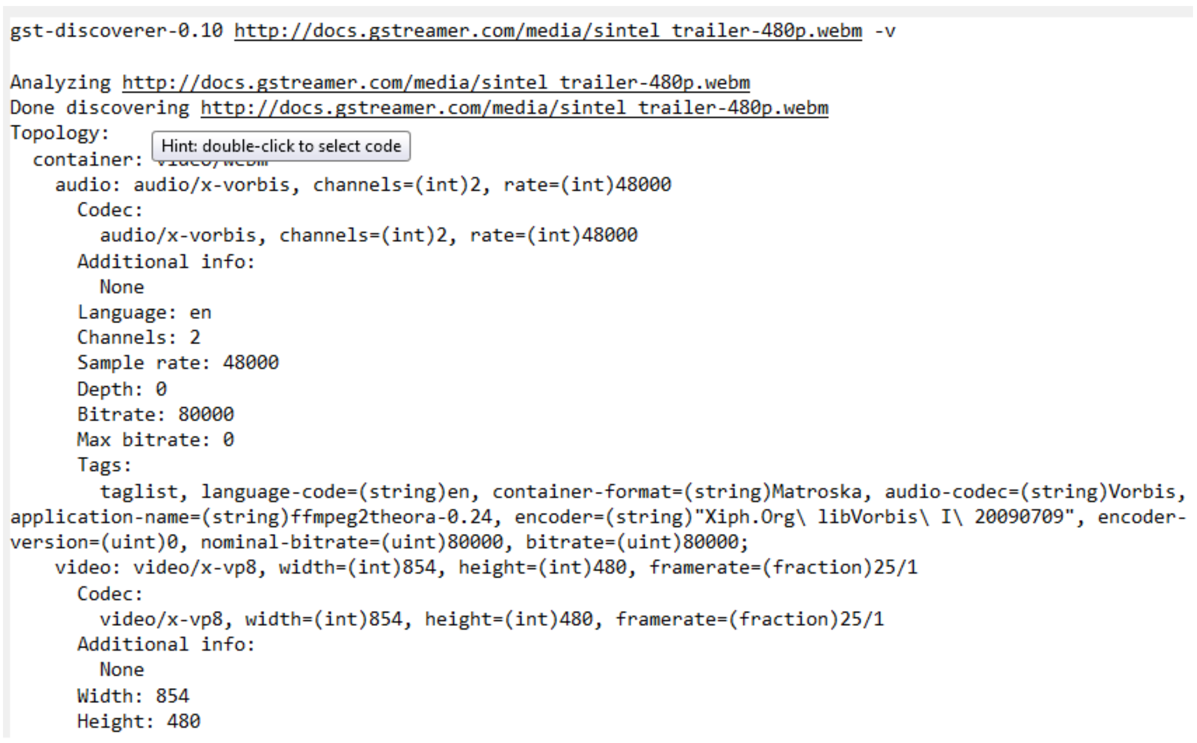
7.4 Work Done : Gstreamer

7.4.1 Compilation of Different Versions of gstreamer

In context of various available versions of Gstreamer compilation of all are done. Compliation of gstreamer-1.X is done

1. Variables are exported.
2. Configuration and compilation is done for

- Gstreamer-1.X core
- Gstreamer-good-plugings
- Gstreamer-bad-plugins
- Gstreamer-base-plugins



```
gst-discoverer-0.10 http://docs.gstreamer.com/media/sintel_trailer-480p.webm -v

Analyzing http://docs.gstreamer.com/media/sintel_trailer-480p.webm
Done discovering http://docs.gstreamer.com/media/sintel_trailer-480p.webm
Topology:
  container: video/webm
    audio: audio/x-vorbis, channels=(int)2, rate=(int)48000
      Codec:
        audio/x-vorbis, channels=(int)2, rate=(int)48000
      Additional info:
        None
      Language: en
      Channels: 2
      Sample rate: 48000
      Depth: 0
      Bitrate: 80000
      Max bitrate: 0
      Tags:
        taglist, language-code=(string)en, container-format=(string)Matroska, audio-codec=(string)Vorbis,
application-name=(string)ffmpeg2theora-0.24, encoder=(string)"Xiph.Org\ libVorbis\ I\ 20090709", encoder-
version=(uint)0, nominal-bitrate=(uint)80000, bitrate=(uint)80000;
    video: video/x-vp8, width=(int)854, height=(int)480, framerate=(fraction)25/1
      Codec:
        video/x-vp8, width=(int)854, height=(int)480, framerate=(fraction)25/1
      Additional info:
        None
      Width: 854
      Height: 480
```

Figure 7.9: gst-discoverer of Gstreamer

7.4.2 Mock Environment Set Up: For Different Branches

Porting from one branch to another cannot be done abruptly, it is required to be tested each and every change in comparison with previous branches. For done there is a need to set up mock environment for different branches so it can be tested along with original branch on which work is going on, whose environment is not on mock.

- There are many branches for which MOCK environment are prepared. These are:
  1. SDK2-10
  2. SDK2-12
  3. SDK2-13
- Steps followed to create MOCK environment are:
  - repo init tag-name-with-branch-name

  - repo sync
  - make a MOCK-ID by running a special script
  - set up environment by using MOCK-ID like:
    ./environemnt.sh -m MOCK-ID
  - make

7.5 Innovations Done

7.5.1 Developing Application : Media Player

As Gstreamer is a media framework or a full fledged library application just need to link from it.It provide built in plugins which come under different categories like good, bad, ugly etc. In these plugins element are provided which are native and can be used by anyone to prepare any type of application.
Steps followed to prepare an APPLICATION :

- Declare elements which we want to use : these names are the names given by user not by gstreamer. For Media Player these elements are used:
  1. Source
  2. Demuxer
  3. Decoder
  4. Sink

- By using Gstreamer function element names are used which provide a unique name to all elements. Elements names used for gstreamer native elements are:
  1. Source : filesrc
  2. Demuxer : oggdemux
  3. Decoder1 : video
  4. Decoder2 : audio

5. Sink1 : fakesink
6. Sink2 : autoaudiosink

- These elements are combined in a bin.
- These are linked. But there is a problem with demuxer that it create dynamic pads, for that a callback is set which on creation of dynamic pads send a signal. In this callback sinkpad for following elements are created and then linked with dynamic pad of demuxer.
  These is done for audio and video decoder.
- After linking state of pipeline is set to PLAYING. Before PLAYING pipeiline go through 3 states. There are 4 states of pipeline
  - NULL
  - READY
  - PAUSE
  - PLAYING
- After that some wait is introduced according to the time we want to play.
- Pipeline state is set to NULL then and pipeline is derefrenced.

7.5.2 Developing Application : Display Bus Messages along with SEEK query enable

By using previous application new function are introduced.

- A bus is introduced on pipeline which display all messages send by elements and pipeline itself.
- By this all messages related to state changes send by pipeline can be seen visually.

- Types of messages send by BUS are:
  - EOS : End of stream
  - STATE-CHANGED :
    1. NULL to READY
    2. READY to PAUSE
    3. PAUSE to PLAYING
  - Any FATAL ERROR
- A query is send to pipeline for obtaining current position and stream duration. Whichever element is able to handle this query it will send the respective values.
- This values are used for these purposes:
  - To find is stream is SEEKABLE.
  - To find SEEK RANGE
  - To SEEK(JUMP) to a specific position.
- So, APPLICATION is ready to send messages and required information

7.5.3 Application : Copy Command By Gstreamer

Simple copy command is made using gstreamer. With the help of this application we can copy a stream into any file.

- A native source is used and in its location property, command line argument is passed as a stream which we want to copy into another file.
  gst-element-set-location(filesrc, "location", argv[1], NULL)
- A native sink is used and name of file in which we want to copy is passed as argument.
  gst-element-set-location(fakesink,"location", argv[2], NULL)

- So this at the end is executed by:
  gst-apps stream-name file-name

7.6 Solved BUGS

There are some issues in existing code which need to be solved. Few issues are reported related to:
1. Catch-live in DVR
2. Circular Buffer

7.6.1 Issue 1 : Catch-live in DVR

In DVR when recording get started then there are few condition to catch-live.
1. Direct catch live by @(at the rate)
2. Catch Live by Forward Trickmode
3. Catch Live by Out of Index Range.

- Problem Statement : When recording is done via TIMESHIFT then if recording is done atleast for 2 minutes the on applying trickmode catch live is observed after few seconds but this is considerable
  But when recording is done for near about 30 seconds only then on applying trickmode, "CATCH-LIVE" is not observed instead of that trickmode goes on continuously.

- Behaviour of Elements :

  - When Catch live is there :
    - DVR-element—sinkevent—-demuxer—sinkevent—-decoder—: get CATCH LIVE event.
  - On Applying Trickmode :

        * decoder—srcevent—-demuxer—sinkevent—-DVR-element—: get SEEK event

- Solution : In demuxer on failure of querry when catch live is there that condition is changed.

7.6.2 Issue 2: Circular Buffer SEEK to '0'

In case of Circular Buffer on SEEK to '0' an issue is observed.

- Problem Statement : On SEEK to '0' Catch Live is observed instead of getting BOS.

- Solution Adopted : Catch Live condition is modified.

    - Earlier: Switch to catch live is there when
        1. Read pointer approaches Write.
        2. Write pointer approaches Read
    - Adopted :

      Switch to catch live is removed when Write pointer approaches Read, instead of that jump to offset is set.

7.7 Summary

This include all details about media framework i.e. gstreamer. It also includes different tools of gstreamer. How to extract debugging information is also included in this. Innovation work include bug solving in media application and made different applications using gstreamer.

Chapter 8

Automation of TestFramework

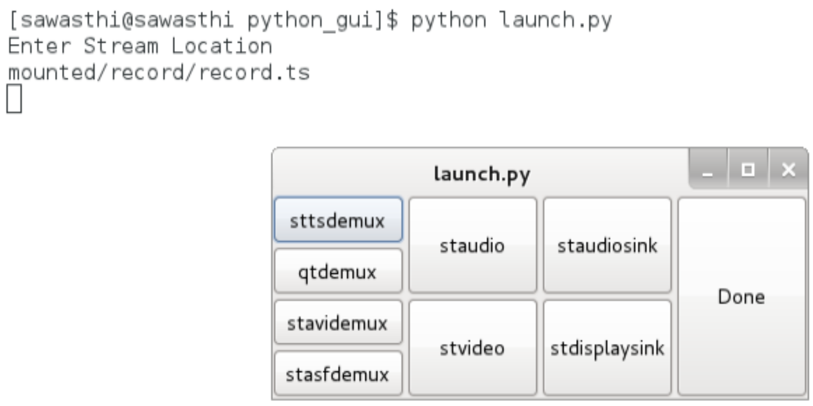
8.1 Need of Automation

As there is a need to do regression testing of already developed code in order to check that its functionality is not breaking. For that purpose if testing is automated then it will check much less time to complete it. Few scripts are made to serve the purpose.

8.1.1 Creating a GUI for gst-launch

A GUI is prepared in order to create a pipeline by connecting different elements provided. This GUI will automatically create a pipeline which will be executed by debugging tool provided by Gstreamer i.e. gst-launch. To do the same PYGTK is used.

- Step 1: Provide Stream Location, which will be set as location property of source.
- Step 2: As soon as location is entered, a GUI is displayed containing elements (as shown in fig 8.1).
- Step 3: Select Elements and make whole pipeline using gst-launch and then it will start playing (as shown in fig 8.2).



```
[sawasthi@sawasthi python_gui]$ python launch.py
Enter Stream Location
mounted/record/record.ts
```

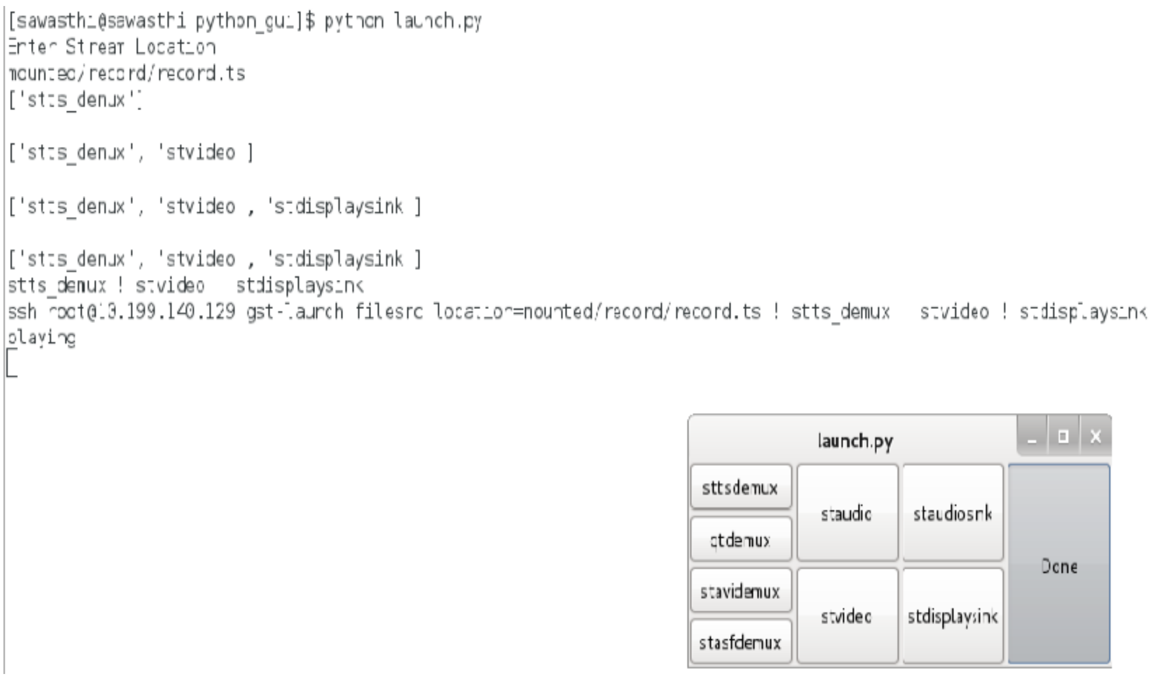
Figure 8.1: GUI

- Benefit: By this script need to remember the name of element is removed and just selected elements get connected one by one and pipeline get executed.

8.1.2 Creating a GUI for LSST/FSST

These tests include around 74 usecses in LSSt and around 700 usecses in FSST. If it will be manual then it will consume lots of time in order to execute all these test cases. So a script is developed in which first all files will be displayed and after that all usecases of selected file will be displayed. We just have to select all usecases which we have to execute and then it will start running one by one. A complete GUI is prepared for the same using SHELL script.

- Step 1: As shown in Fig. 8.3, a prompt is displayed to select the file.
- Step 2: Select the file whose usecases we want to play(as shown in fig. 8.4)
- Step 3: Select the usecases(as shown in fig. 8.5). Multiple usecases can also be selected.
- Enter the TARGET IP on which we want to play, as shown in fig. 8.6.
- Benefit : When particular use case is required to be run then this is most beneficial.



```
[sawasthi@sawasthi python_gui]$ python launch.py
Enter Stream Location
mounted/record/record.ts
['stts_demux']

['stts_demux', 'stvideo ]

['stts_demux', 'stvideo , 'stdisplaysink ]

['stts_demux', 'stvideo , 'stdisplaysink ]
stts_demux ! stvideo   stdisplaysink
ssh root@13.199.140.129 gst-launch filesrc location=mounted/record/record.ts ! stts_demux   stvideo ! stdisplaysink
playing
```

Figure 8.2: Executing Pipeline by gst-launch

8.1.3 GUI for providing UserOps

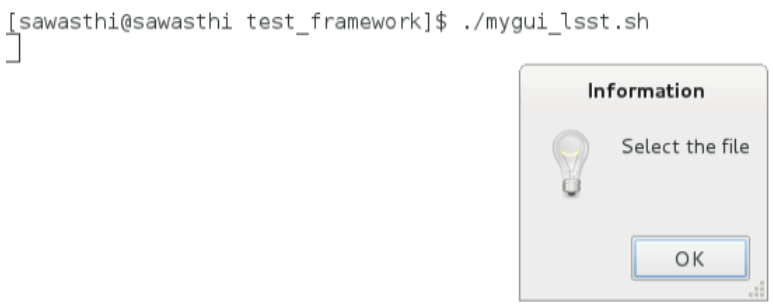
It is difficult to run each and every case manually. So all options are provided to run all usecases from anywhere.

- Step 1: As shown in fig 8.8, enter the source from where we want to get the stream
- Select the options in sources.
- Select Userops we want to apply, as shown in fig 7.10.
- Whole Pipeline will be connacted by gst-apps

8.1.4 Coverity Automation

It was needed to automate coverity to make it easier to run.

- Select the module(as shown in fig. 8.12).



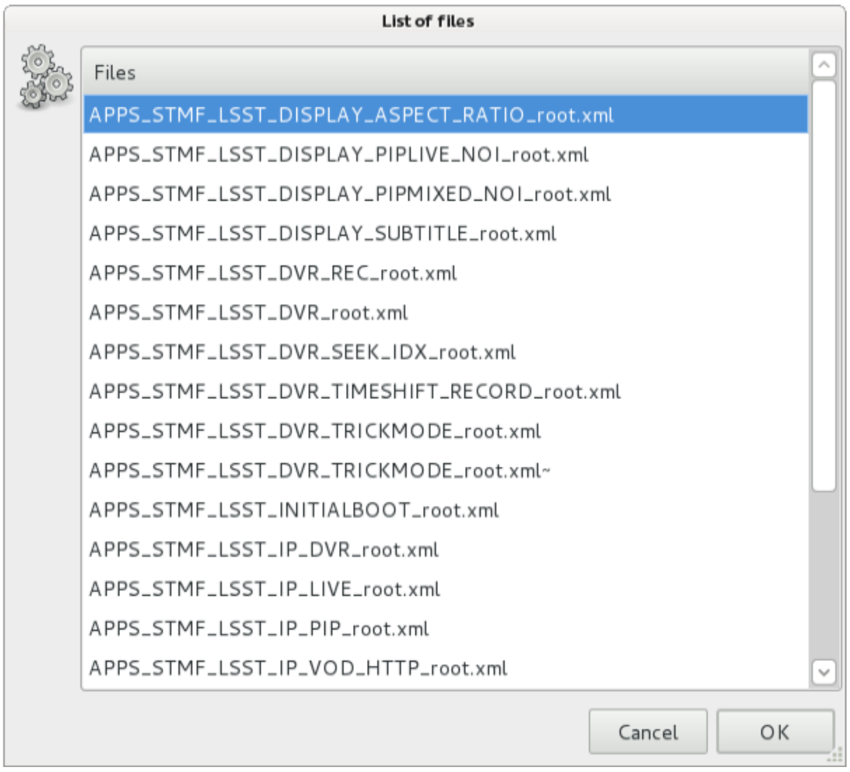
```
[sawasthi@sawasthi test_framework]$ ./mygui_lsst.sh
```

Figure 8.3: File Selection Prompt

- Select and run all the selected modules.
- see the results by viewing html file, as shown in fig 8.13.

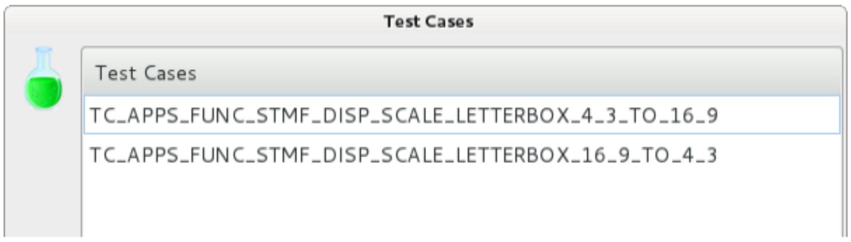
8.2 Summary

For running such a large test suites, it is required to automate these test suites so that whole procedure will be easily executed. Along with that it is difficult to build a pipeline manually using gst-launch because we have to remember name of all elements. By using gui it will be automatically executed on target. Automation of coverity tool is also a helping hand in improving the work of user. Innovation work include gui made for gst-launch, gui for userops, gui for coverity tool etc.



APPS_STMF_LSST_DISPLAY_ASPECT_RATIO_root.xml
APPS_STMF_LSST_DISPLAY_PIPLIVE_NOI_root.xml
APPS_STMF_LSST_DISPLAY_PIPMIXED_NOI_root.xml
APPS_STMF_LSST_DISPLAY_SUBTITLE_root.xml
APPS_STMF_LSST_DVR_REC_root.xml
APPS_STMF_LSST_DVR_root.xml
APPS_STMF_LSST_DVR_SEEK_IDX_root.xml
APPS_STMF_LSST_DVR_TIMESHIFT_RECORD_root.xml
APPS_STMF_LSST_DVR_TRICKMODE_root.xml
APPS_STMF_LSST_DVR_TRICKMODE_root.xml~
APPS_STMF_LSST_INITIALBOOT_root.xml
APPS_STMF_LSST_IP_DVR_root.xml
APPS_STMF_LSST_IP_LIVE_root.xml
APPS_STMF_LSST_IP_PIP_root.xml
APPS_STMF_LSST_IP_VOD_HTTP_root.xml

Figure 8.4: File Selection



TC_APPS_FUNC_STMF_DISP_SCALE_LETTERBOX_4_3_TO_16_9
TC_APPS_FUNC_STMF_DISP_SCALE_LETTERBOX_16_9_TO_4_3

Figure 8.5: Use Case Selection

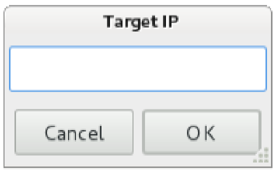
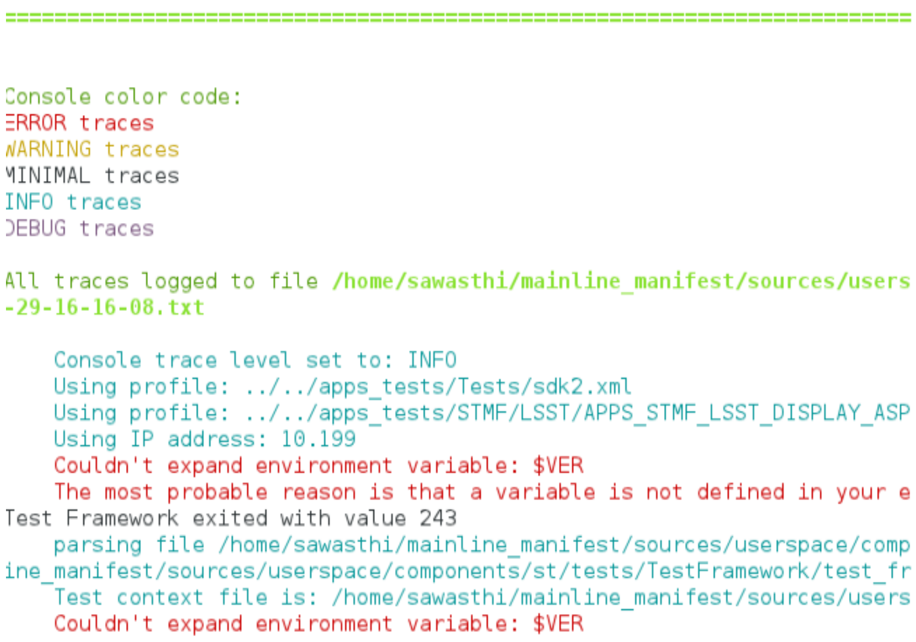


Figure 8.6: Entering Target IP



```
==========================================================================


Console color code:
ERROR traces
WARNING traces
MINIMAL traces
INFO traces
DEBUG traces

All traces logged to file /home/sawasthi/mainline_manifest/sources/users
-29-16-16-08.txt

    Console trace level set to: INFO
    Using profile: ../../apps_tests/Tests/sdk2.xml
    Using profile: ../../apps_tests/STMF/LSST/APPS_STMF_LSST_DISPLAY_ASP
    Using IP address: 10.199
    Couldn't expand environment variable: $VER
    The most probable reason is that a variable is not defined in your e
Test Framework exited with value 243
    parsing file /home/sawasthi/mainline_manifest/sources/userspace/comp
ine_manifest/sources/userspace/components/st/tests/TestFramework/test_fr
    Test context file is: /home/sawasthi/mainline_manifest/sources/users
    Couldn't expand environment variable: $VER
```

Figure 8.7: Executing LSST

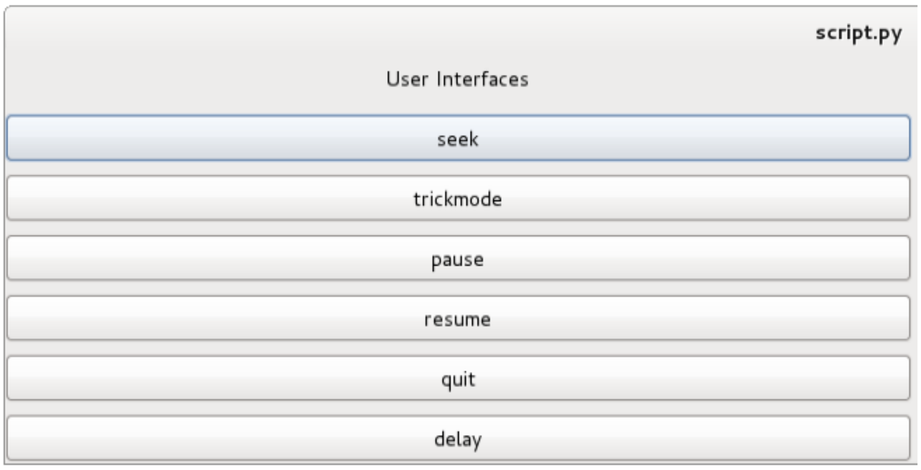


Figure 8.8: Source options

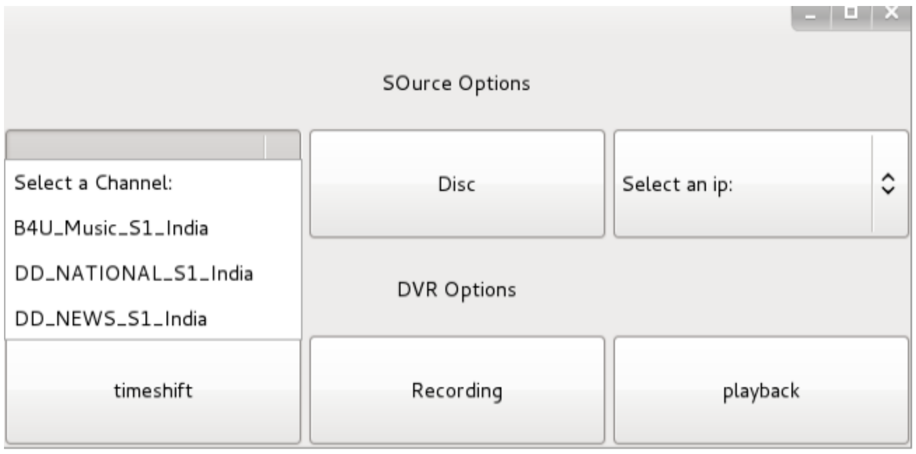


Figure 8.9: Available Channels

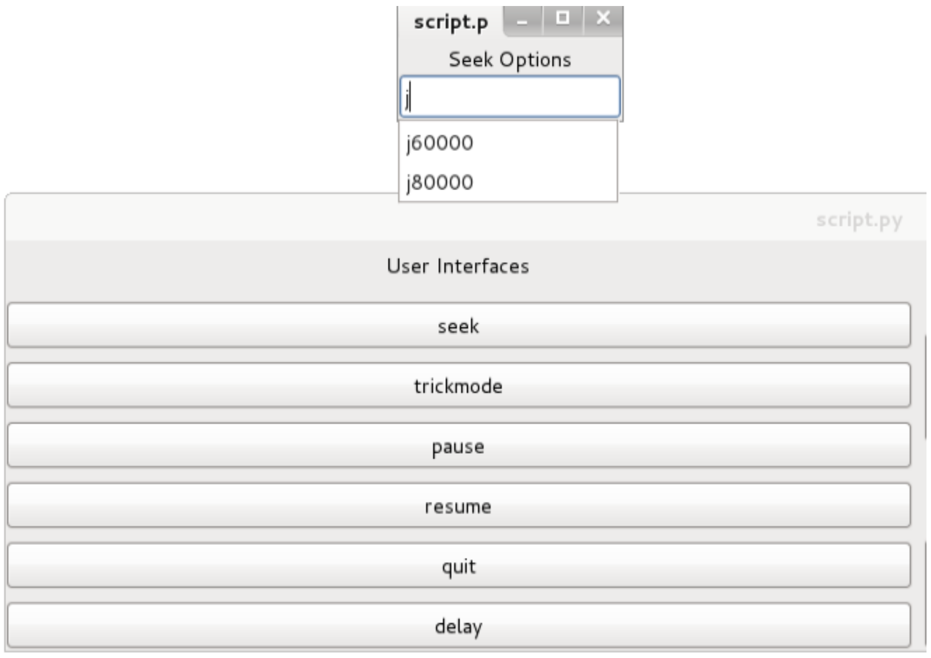
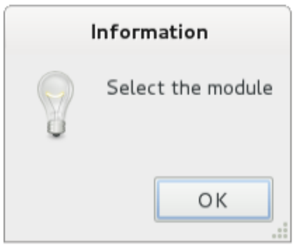
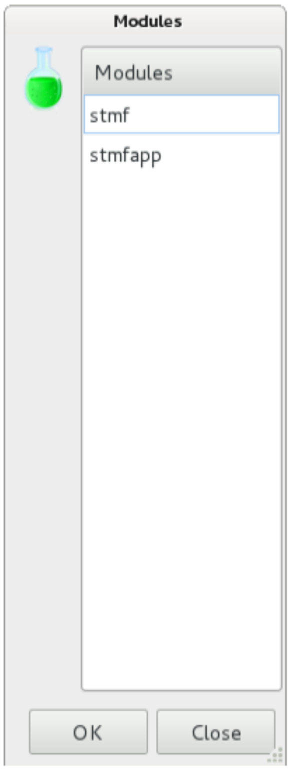


Figure 8.10: User Ops option



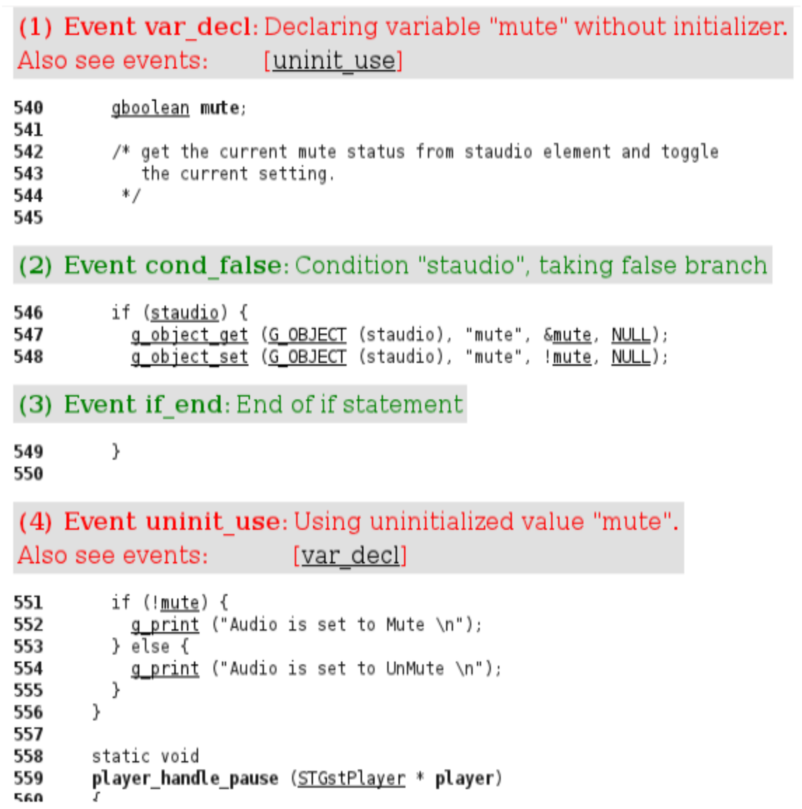
Select the module

Figure 8.11: Information about module



stmf
stmfapp

Figure 8.12: Select Modules



(1) Event var_decl: Declaring variable "mute" without initializer.
Also see events: [uninit_use]

```
    gboolean mute;

    /* get the current mute status from staudio element and toggle
       the current setting.
     */

```

(2) Event cond_false: Condition "staudio", taking false branch

```
    if (staudio) {
      g_object_get (G_OBJECT (staudio), "mute", &mute, NULL);
      g_object_set (G_OBJECT (staudio), "mute", !mute, NULL);
```

(3) Event if_end: End of if statement

```
    }

```

(4) Event uninit_use: Using uninitialized value "mute".
Also see events: [var_decl]

```
    if (!mute) {
      g_print ("Audio is set to Mute \n");
    } else {
      g_print ("Audio is set to UnMute \n");
    }
  }

  static void
  player_handle_pause (STGstPlayer * player)
```

Figure 8.13: Result of Coverity

Chapter 9

Conclusion and Future Work

9.1 Conclusion

It is must to do regression testing of code because in open source community continuouslybdevelopment is going on and new changes are incorporating every minute.So, it is required to check that old functionalities are not breaking.
To do the same with less efforts and save time there is a need of automation. Automation is done by scripting (Shell and Python). By automation manual work is converted in GUIs. Executing any test is just a button press away with the help of these GUIs.
DVR is the need of user when it comes to set top box. New improvements are going on continuously. Circular buffer is one of them. Along with various available features in DVR circular buffer keeps record of any program according to the buffer size provided, it provide last recorded program accoring to size i.e. it keeps on overwriting even when buffer gets filled. This Circular buffer is required to be studied and tested. It has certaing bugs related to catch live issue which were required to be resolved with refrence to user requirement. Gstreamer which is a multimedia framework supporting all types of applications in open source environment is the one used for building all media apps so any issue in its code used to be resolved as urgent as

possible. So continuous efforts are made to do the same.Various tools are provided by gstreamer which are helpful for different purposes like in debugging(gst-launch), element-information(gst-inspect) and media information(gst-discoverer) etc. So these are required to be studied and worked on.
Along with that bugzilla and Gerrit are the tools which are used to lock the bug. Gerrit is the one which is replacing the existing bugzilla. These must be studied.

9.2 Future Work

- BER improvement with Combination of FEC and MDC in heterogeneous environment:
  - Combination of FEC along with MDC with better deciding factors make it one of the best scheme suitable for any heterogeneous environment.
  - Further study of HEVC will led to better scheme for route selection
- Automation led to automatic verdict generation and give reports. A press of few buttons will complete whole regression testing.
- New features support can be provided using Gstreamer. Ex: Change of video in main and pip.
- New plugins can be made which make linking much easier and provide more properties to be utilized.
- More Scope of improvements in Circular Buffer.
- Gerrit can be the new tool providing better support than bugzilla.
- Existting issues can be resolved to improve functionality of build in apps.
- Improvement in automation of coverity tool.

References

[1] ST Internal Modules, "Introduction to Set Top Boxes"

[2] II-Koo Kim, Junshye Min, Tammy Lee, Woo-Jin Han, and JeongHoon Park, Block Partitioning Structure in the HEVC Standard, IEEE Transactions on Circuits and Systems for video technology, vol 22, no. 12, December 2012

[3] Jens-Rainer Ohm, Gary J, Sullivan, Heiko Schwarz, Thiow Keng Tan, Thomas Wiegand, Comparison of the Coding Efficiency of Video Coding Standards-Including High Efficiency Video Coding (HEVC), Pre-Publication Draft, to appear in Transactions on Circuits and Systems for video technology, December 2012.

[4] ST Internal Document,"Introduction to Test Framework"

Publications

Title : "BER Improvement With Combination of FEC and MDC in an heterogeneous Environment"

Authors : Preeti Dewani, Ankit M. Prajapati, Bhavin Patel

Conference : ST Tech Week, 2014

Abstract : FEC(Forward Error Correction) serves the purpose of error correction mainly in the case of small packet length, high redundancy and low packet loss rate but when it comes to large packet length, small redundancy and high packet loss rate then MDC (Multiple Description Coding) is preferable. Due to network diversity and end-user device diversity there is a need of scalability. MDC outperforms FEC especially when it comes to large packet length or in other words, in case of high bandwidth utilization with minimum network knowledge. In high packet loss condition and congestion in the network, MDC is the solution to solve the problem. Along with MDC, FEC usage is the new proposal which is also needed in case when burst length is small and high redundancy is there.

The proposal is that in case when we have high redundancy and small packet size FEC shows better performance than MDC but when it comes to effective bandwidth utilization by means of large packet size and low redundancy FEC performances sharply degrades while MDC shows better performance. So, the combination of both when applied in heterogeneous channel will lead to: 1) BER improvement. 2) Effective bandwidth utilization. 3) Higher recovery rate. 4) Scalability. 5) Graceful degradation in case of high packet loss rate. 6) Ease of recovery by FEC(as higher redundancy is here) decreases congestion and improves QoS. 7) Better PSNR will be obtained when packet loss is high, using MDC. 8) Adaptability to heterogeneous environment.