Interactive Set-top Box: Software Integration and Validation

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

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

By

Abhay Maheta
(13MECE25)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Interactive Set-top Box: Software Integration and Validation

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

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

By

Abhay Maheta
(13MECE25)

Under the guidance of

External Project Guide:

Mr. Vaibhav Pathak
Senior Project Manager,
ST Microelectronics Pvt. Ltd.,
Greater Noida.

Internal Project Guide:

Prof. Yogesh Trivedi
EC Department,
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Declaration

This is to certify that

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

b. Due acknowledgment has been made in the text to all other material used.

- Abhay Maheta

CERTIFICATE

This is to certify that the Major Project entitled ***"Interactive Set-top Box: Software Integration and Validation"*** submitted by **Maheta Abhay M. (13MECE25)**, towards the partial fulfillment of the requirements for the degree of **"Master of Technology" in *"Embedded Systems"*** of **Nirma University of Science & Technology, Ahmedabad** is the record of work carried out by him under our supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for the examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: _____

Place: Ahmedabad

Internal Guide:

_____
Prof. Yogesh Trivedi
Sr. Associate Professor
Nirma University

PG Co-ordinator

_____
Dr. Nagendra Gajjar
Embedded Systems
Nirma University

Director:

_____
Dr. K. Kotecha
Director, IT-NU

HOD:

_____
Dr. P. N. Tekwani
Head of EE Dept.

Certificate

This is to certify that the Major Project entitled **"Interactive Set-top Box: Software Integration and Validation"** submitted by **Abhay Maheta(13MECE25)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

Mr. Vaibhav Pathak
Senior Project Manager
ST Microelectronics Pvt. LTD
Greater Noida

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr. P.N.Tekwani**, Head of Electrical Engineering Department, and **Dr. N.P.Gajjar**, PG Coordinator of M.Tech Embedded Systems program for allowing me to undertake this thesis work and for his guidelines during the review process.

I take this opportunity to express my profound gratitude and deep regards to **Prof.Yogesh Trivedi**, guide of my major project for his exemplary guidance, monitoring and constant encouragement throughout the course of this thesis. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I would take this opportunity to express a deep sense of gratitude to Project Manager **Mr. Vaibhav Pathak** ,Senior Project Manager, ST Microelectronics Pvt Ltd. for his cordial support, constant supervision as well as for providing valuable information regarding the project and guidance, which helped me in completing this task through various stages.

I would also thank to **Mr.Manish Sharma**,my Project Mentor for always helping,give good suggestions and solving my doubts and guide me to complete my project in better way.

I am obliged to **Preeti Verma** staff member of ISTB team, ST Microelectronics Pvt. Ltd. for the valuable information provided by her in her respective fields. I am grateful for her cooperation during the period of my assignment.

Lastly, I thank almighty, my parents, brother, sisters and friends for their constant encouragement without which this assignment would not be possible.

- Abhay Maheta
13MECE25

Abstract

In recent years with advent of digital era, technologies like Digital Television, Internet, Local Area Network, IP telephony have evolved the concept of Digital Home. Interactive set-top box, also called Headed Gateway fulfills need of residential gateway, digital set-top box and telephony as a Complete Solution. Here in this thesis such a Home gateway system is presented which has residential gateway with IP telephony as Front End and Digital set-top box as Back End. Gateway is cable modem based device (Euro-DOCSIS) which is dual CPU architecture, Cable modem and ARM core. Digital set-top box has all audio/video decoding capabilities. Both are self-sufficient and independent enough. This whole system has Linux based operating system with kernel 3.10.

This thesis includes integration and full software system validation of Interactive Set-top box (ISTB). Integration over here means to unite together various software components along with kernel, with valid release and build complete software of ISTB. Full system software validation is process of being assured that product is meeting specified requirements. Thus ISTB software validation means to check functionalities like Linear TV use case, Local Media Playback, Data Browsing, etc.

Sanity testing is a type of software validation process which is performed after verification of software and assures previous bugs are resolved and working funcionality is not broken. Software integration of ISTB is similar to sanity testing where software components with new releases are integrated and get surity of solution of previous limitations. Full system validation is post sanity process where when a full system performance is validated with all specified requirements i.e. use cases related to ISTB.

ISTB use cases are classifies as:

- Functional
- Parallel
- Stress
- Robustness

Full system software validation is burdensome, time consuming and more prone to human error. Automation of such testing using Python and XML language makes validation process efficient and less complex.

Company Profile

ORGANISATION PROFILE:

STMicroelectronics is the worlds fifth largest semiconductor company with net revenues of US$8.51 billion in 2009. Offering one the industrys broadcast product portfolios, ST serves customers across the spectrum of electronics applications with innovative semiconductor solutions by leveraging its vast array of technologies, design expertise and combination of intellectual property portfolio, strategic partnerships and manufacturing strength. STMicroelectronics was created in 1987 by the merger of SGS Microelectronic of Italy and Thomson Semiconductors of France with the aim of becoming a world leader in the sub-micron area. The new company pursued on aggressive growth strategy, investing heavily in R&D, forging strategic alliances with blue-chip customers and academia, building up an integrated presence in major economic regions, and honing one of the worlds most efficient manufacturing operations.

According to the latest industry data, ST is the worlds fifth largest semiconductor company with market leadership in many fields. For example, ST is the leading producer of application-specific analog chips and power conversion devices. It is also the #1 supplier of semiconductors for the Industrial market and for set-top-box applications, and occupies leading positions in fields as varied as discrete devices, camera modules for mobile phones and automotive integrated circuits.

CORPORATE RESPONSIBILITY:

STMicroelectronics was one of the first global industrial companies to recognize the importance of environmental responsibility and, over the past 15 years, the Companys sites have received more than 100 awards for excellence in all areas of Corporate Responsibility, from quality to corporate governance, social issues and environmental protection. The Companys corporate responsibility policy is detailed in its Principles for Sustainable Excellence.

PRODUCT PORTFOLIO:

ST offers one of the worlds broadest product ranges, with over 3,000 main types of products. The carefully balanced portfolio includes both application-specific products containing a large proprietary IP content and multi-segment products that

range from discrete devices to high-performance microcontrollers. St pioneered and continues to refine the use of platform-based design methodologies for complex ICs in demanding applications such as set-top-boxes, secure smart cards and mobile multimedia, which minimizes development time and cost. The balanced portfolio approach allows ST to address the needs of all microelectronics users, from global strategic customers for whom ST is the partner of choice for major System-on-chip (SoC) projects to local enterprises that need fully-supported general-purpose devices.

MANUFACTURING MACHINE:

To provide its customers with an independent, secure a cost-effective manufacturing machine, ST operates a worldwide network of front-end (wafer fabrication) and back-end (assembly and test and packaging) plants. STs principal wafer tabs are presently located in Agrate Brianza and Catania (Italy), Crolles, rousset and Tours (France), and Singapore. The wafer tabs are complemented by world-class assembly-and-test facilities located in China, Malaysia, Malta, Morocco and Singapore.

RESEARH AND DEVELOPMENT:

Since its creation, ST has maintained an unwavering commitment to R&D and in 2009 it spent US$2.37 billion I n R&D, which is approximately 28% of its revenue, and includes the R&D activities related to ST Ericsson, as consolidated by St. Among the industrys most innovative companies ST draws on a rich pool of chip fabrication technologies, including advanced CMOS (Complementary Metal Oxide Semiconductor), mixed-signal, analog and power processes, and is a partner in the International Semiconductor Development Alliance (ISDA) for the development of next-generation CMOS technologies.

THE KNOWLEDGE NETWORK:

ST has developed a worldwide network k of strategic alliances, including product development with key customers, technology development with customers and other semiconductor manufacturers, and equipment-and Cad-development alliances with major suppliers. These industrial partnerships are complemented by a wide range of research programs conducted with leading universities and research institutes around the world. By augmenting its rich portfolio of proprietary technologies and core competencies with complementary expertise from a variety of carefully chosen strategic partners, ST has developed an unsurpassed capability to offer leading-edge solutions to consumers in all segments of the electronics industry.

Many of STs research and development programs are managed by its AST (Advanced System Technology) organization, whose mission is to develop the strategic

system knowledge that will be required within 3-5 years by Sts product divisions. Among ASTs significant recent achievements are innovative technologies for digital consumer, networking, mobile security, and on-chip interconnect.

SUSTAINABLE EXCELLENCE:

STs technical, marketing, and manufacturing strengths are matched and further enhanced by au unswerving commitment to Sustainable excellence that has earned prestigious awards around the world. Since 1991, the Companys sites have received more than 70 awards for excellence in all areas of Corporate Responsibility, from quality to corporate governance, social issues a d environmental protection.

STs commitment to environmental responsibility has resulted in substantial reductions over the years in the consumption of energy, water, paper, and hazardous chemicals, increased recycling of waste products and a significant cut in greenhouse-gas emissions. St has constantly pushed the boundaries of excellence in Corporate Responsibility, achieving outstanding performance in key areas such as occupational health and safety- including the certification of 16 manufacturing sites and 4 non-manufacturing sites to the international standard OHSAS 18001; the application of low-power technology to its wad e product range; an bridging the digital divide through the Digital Unify Program, led by the STMicroelectronics Foundation.

FACTS AND FIGURES:

The group totals approximately 50,000 employees, 16 advanced research and development units, 39 design and application centers, 17 main manufacturing sites and 78 sales offices in 36 countries. Corporate Headquarters, as well as the head quarters for Europe and for Engineering Markets, are in Geneva. The Companys U.S. Headquarters are in Carrollton (Texas); those for Asia-Pacific are based in Singapore and Japanese operations are headquarters in Tokyo. The recently- established Greater China region, which includes Hong Kong, China and Taiwan, is head quartered in Shanghai.

The Company now has around 900 million outstanding shares, 72.4% of which are publicly traded on the various stock exchanges. The balance of the shares is held by STMicroelectronics Holding II B.V., a company whose shareholders are CassiaDeposite Prestiti and Finmeccanica of Italy, and Areva of France.

Contents

List of Figures

List of Tables

Acronyms

API ..... Application Programming Interface
ADSL ..... Asymmetric digital subscriber line
BE ..... Back End
BER ..... Bit Error Rate
CA ..... Conditional Access
CEL ..... Cable Europe Labs
DVB ..... Digital Video Broadcasting
DVR ..... Digital Video Recording
DVD ..... Digital Versatile Disc
ECL ..... EuroCableLabs
ES ..... Elementary Stream
FE ..... Front end
FEC ..... Forward Error Correction
GW ..... Gateway
HDMI ..... High Definition Multimedia Interface
HFC ..... Hybrid Fiber Coax
I2C ..... Inter-Integrated Circuit
IR ..... Infra-Red
ISTB ..... Interactive Set-top Box
ISO ..... International Standards Organization
JTAG ..... Joint Test Action Group
MPEG ..... Moving Picture Experts Group
MPTS ..... Multi Program Transport Stream
MSPS ..... Mega Symbols Per Seconds
NIM ..... Network Interface Module
OS ..... Operating System
PES ..... Packetized Elementary Stream

PID ..... Packet Identifier
PS ..... Program Stream
PSI ..... Program-specific Information
PSTN ..... Public switched telephone network
RTOS ..... Real Time Operating System
SoC ..... System on Chip
SPI ..... Serial Peripheral Interface
SPTS ..... Single Program Transport Stream
SSH ..... Secure Shell
STMC ..... ST Micro Connect
TS ..... Transport Stream
UART ..... Universal Asynchronous Receiver/Transmitter
USB ..... Universal serial bus
VCS ..... Version Control System

Chapter 1

Introduction

1.1 Motivation

Electronic products have become living requirement of men kind. Such as television has become such vital part of home in modern world that it is hard to imagine life without television. Television has emerged from analog to digital & mono color to Ultra High Definition. Definition of **Set-top Box** can be given as a digital appliance which contains tuner as input and output is displayed on connected Television and converting source signal into such a form that can be presented on television screen. A **gateway** is:

- Gateway is a proxy server or router that creates a bridge between two networks
- Gateway Rule - Connected client and Gateway must be in same subnet

In modern era a **Set-top box** and **Gateway** (generally called modem) are most common home entertainment electronics product found. So instead of having two different electronic products, a **complete solution** would be a smarter move. This complete solution is Interactive Set-top Box(**ISTB**).

ISTB gives user the flexibility of interaction with two Worlds (TS based Linear TV and Internet-based Operator Services). It is having a gateway along with IP telephony in Front-end(FE) and Set-top box capabilities in Back-end(BE). Both are self-reliant and independent. Thus it gives a smart unification of two most common home entertainment gadgets.

Validation process with respect to software project management can be defined as process of analyzing that software product or system delivers the result which is fulfilling all specified requirements and purposes. It might be broached as **software quality control**.

1.2 Thesis Objective

Main aspiration of the thesis is Integration and Validation of software system of Interactive Set-top Box,

- High level examination or checking of software product about meeting customer's requirement or specifications

Software validation is process of evaluation of software at the end of or during development to find specified requirements are met or not.
Moreover validation process covers ratification of following use cases:

- Linear TV
- ZAP
- Trick Mode
- MediaPlayer
- Digital Video Recording(DVR)
- Data Browsing (Wired & Wireless)

- Video Streaming

To write script for Sanity testing and System testing(explained later) in automated environment.

1.3 Thesis Structure

Thesis structure is as follows. Chapter-2 introduces Interactive Set-top Box and explains both Hardware and Software architecture of ISTB. Chapter-3 explains DOCSIS protocol in depth. In chapter-4 a small description of MPEG stream is given where MPEG transport stream and various containers are described. In Chapter-5, a small explanation of version control in software development is given. Chapter-6 gives brief idea of testing and software testing of ISTB, mainly sanity testing. Chapter-7 deals with set up environment required for ISTB testing in both software (Linux) and hardware. Chapter-8 gives in brief the process of software integration and validation of ISTB system and the execution of ISTB sanity and system testing in manual and automated environment.

Chapter 2

System Architecture

Interactive Set-top Box is CM-CMTS based architecture, which provide a return channel and Internet access from HFC network. This system is Data Over Cable Service Interface Specification - DOCSIS 3.0 compliant. Full system is divided into two parts.

- Gateway, code named as Alicante
- Set-top Box, code named as Cannes

Gateway as in front-end is based on Cable modem (CM) based device. All data computation are done at gateway side. Set-top Box in back-end does all multimedia computation. The system maintains total 16x4 channels meaning 16 downstream and 4 upstream channels, of which 8 downstream channel are allocated for video/audio, 8 channels for downstream data and 4 channels for upstream data. Cable Modem Termination System (CMTS) as head-end is located in back bone where data from CMTS and Digital Video Broadcasting (DVB) is merged and Muxed data comes as an input at ISTB tuners.[8] **ISTB** system architecture is explained in figure-2.1. Hardware and Software Architecture of ISTB is explained below.

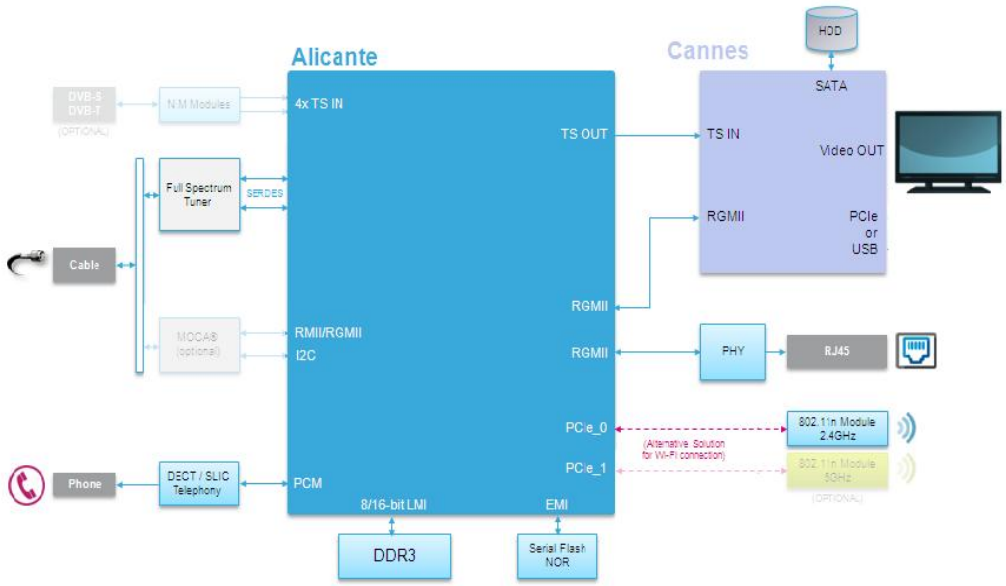


Figure 2.1: ISTB System Architecture[1]

2.1 Hardware Architecture

2.1.1 Gateway

Gateway (Front end) here is Cable Modem based device which extracts row data for Set-top Box (Back end) and Internet packets from Muxed input from CMTS. Cable modem follows Euro-DOCSIS 3.0 specification. It is dual SoC IC which contains ARM Cortex-A9 processor and ST-40, STMicroeclectronics solution as Cable Modem.

Tuner: The very first module that incoming RF streams deal with is tuner. Requested data is extracted according to the tuning.

Power Supply: Requirement to run various hardware components and peripherals is satisfies by power supply. In case of satellite feed device power mechanised antennae might be used to get allignment in different directions.

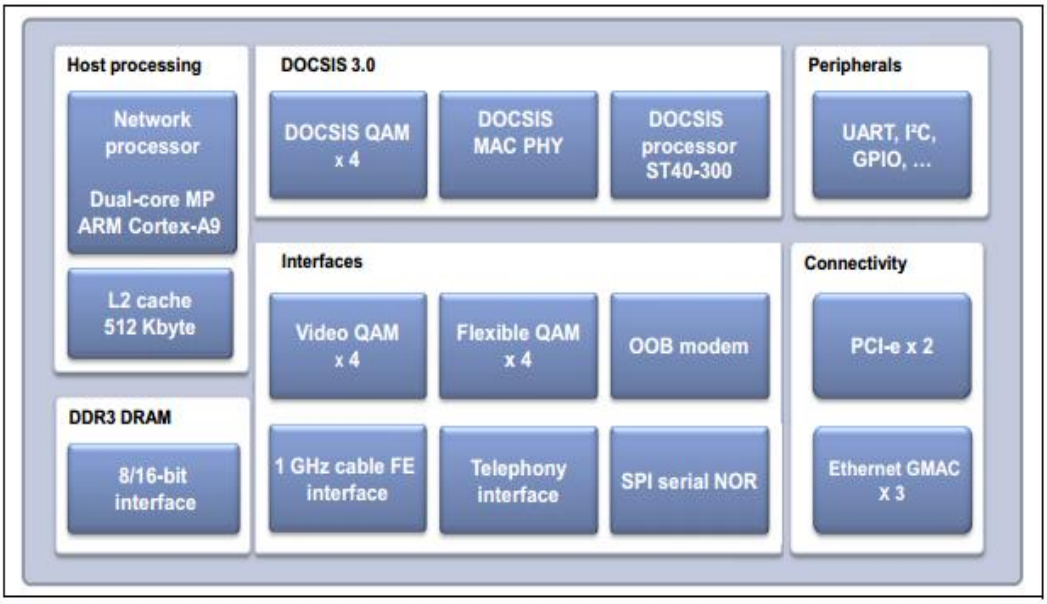


Figure 2.2: Gateway Hardware[1]

CPU: As shown in figure-2.2 ARM Cortex A9 is central processor or host processor. Here dual core processor is used. ARM Cortex A9 is used for low power and cost-sensitive devices. It is designed around dual-issue suparscalar and out-of order dynamic length pipeline (8-11 stages). It's micro-architecture supports 16, 32 or 64KB four way associative L1 caches, with an optional L2 cache providing up to 8MB of cached memory. Working frequency is around 800 MHz as maximum with voltage overdrive. 128 MB DDR3 RAM is connected with it.

Cable Modem: ST-40, STMicroelectronics solution is used as Cable Modem which is Euro-DOCSIS 3.0 compliant. CM as a terminal device handles the transition between TS packet and IP datagram. Initiation of TS packet happens at output port of CMTS which travels through HFC network and termination happens at inout of CM. CM provides two way connectivity between System and server. It supports Quadrature Amplitude Modulation (QAM).

Peripherals: Supplementary components other than CPU, many peripheral devices are supported by ISTB providing additional features.

USB: To store recorded media into externel device

SATA: Same purpose as USB but with higher data transfer speed

Ethernet: For data browsing for connected client

UART: To capture debug prints

I2C: Main CPU uses this to talk with peripherals like front end, display controllers.

SPI: Used to connect not-volatile memory

2.1.2 Set-top Box

Hardware components of Set-top Box are[2]:

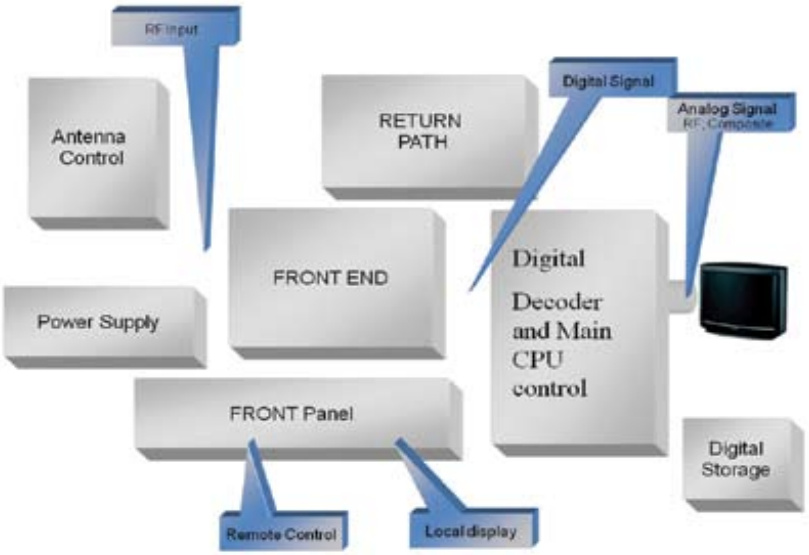


Figure 2.3: Various hardware components inside an STB[2]

Front Panel: It is hardware peripheral device which contains a micro-controller which acts as co-processor and mediocre between CPU and user input. An infrared(IR) receiver continuously listens to remote control. It is situated at outer frame of box that connects some input buttons.

Front end: It is made up of tuners which tunes to particular channel and demodulator which converts RF signal and feeds to decoders in form of transport stream. Technically it is also termed as Network Interface Module (NIM).

A Digital Decoder: Output of front end more specifically of demodulator is received by digital decoder. This input is further demultiplexed and decompressed. The task of the demultiplexer is to separate all streams packaged together. This streams contain video, audio and specified data like Program Specific Information (PSI) tables. MPEG-2 or MPEG-4 format is then achieved by performing decompression by the decoder. Further processing on set of data is done by CPU now like graphical and memory operations.

CPU: It is ARM based CPU same as used in Gateway. Linux based RTOS runs on this PC and it handles all peripherals.

Digital Storage: It is suppliment or optional component inside ISTB. Necessity of storage is for steady storage of all kind of data. Its communication with CPU is controlled by device drivers of RTOS modules.

CA Module: It is categorized under most important component of ISTB. Virtually it is lifeline for all solution providers. It is also known as Integrated Conditional Access Module (ICAM). Placement of this module is done just before the demultiplexer so that encrypted signal can easily be descrambled. There exists an interface for smart card to provide security feature.

Smart Card Slot: Conditional Access implementation is done using this slot to use smart card. Each card stores unique subscriber ID. It also used in decrypting the channels.

Graphics Engine: It is most powerfull co-processor which is dedicated for acceleration of graphics. Purpose of this component is to provide Graphical User Interface(GUI) to customer. After introduction of 3-D menus, this components have evolved more and has become an essential component for ISTB.

Return Path: Here in case of cable modem based device this provides path to communicate with head end. This is helpful in case where customer wants to buy PPV (Pay Per View) directly using credit cards.

Hardware Set up Components

ST Micro Connect: It is used to load compiled image of kernel in host machine to the SoC of development board. It is intermediary device which connects JTAG interface of target board with host PC and gives facility to host PC to boot target board, to download image into target board. The installation and usage of Microconnect is easy. The package supplies software utilities and firmware containg target packs for particular development boards.

Figure 2.4: ST Micro Connect[1]

UART: Debug information is obtained via serial communication between target board and host machine using UART.

JTAG: This intermediary medium gives ease to load kernel image into development board.

Figure 2.5: JTAG[1]

2.2 Software Architecture

ISTB software is divided into two parts as shown in figure-2.6, userspace and kernelspace. Kernelspace includes Linux kernel 3.10, all kernel modules & drivers developed by STMicroelectronics and open source community. Broad Classification of ISTB software is shown in figure-2.7.

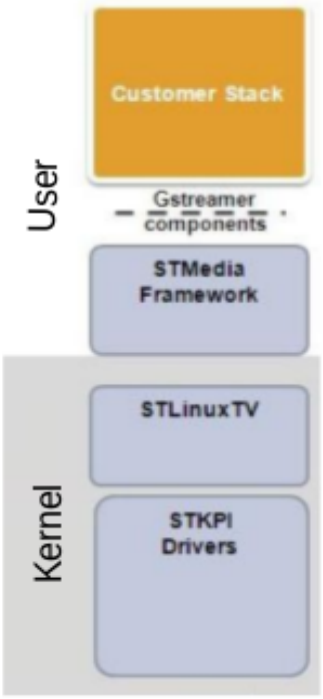


Figure 2.6: Stack View of ISTB Software

2.2.1 KernelSpace

KernelSpace as specified above is made up of Linux kernel and kernel modules. Linux kernel here is kernel version 3.10 which is most widely used in embedded systems. It is responsible for all process management, memory management and security. Of many kernel modules developed by STMicroelectronics some are explained below:

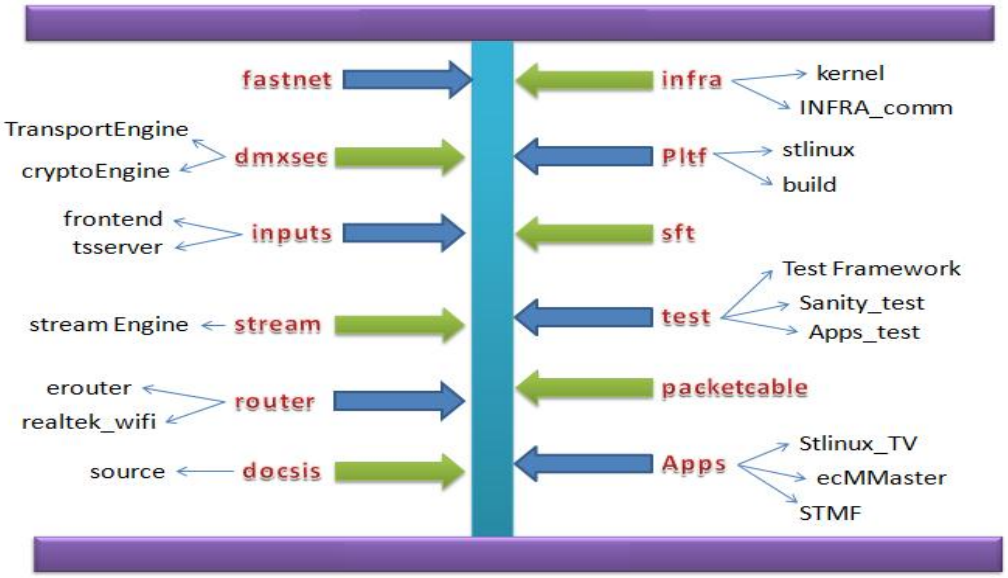


Figure 2.7: Classified view of ISTB software

Frontend Engine: The Frontend Engine (FE) component wraps all of the functionalities performed by different digital, analog tuner/demod sources, and IP tuner sources. The Frontend Engine is the top level component and it exposes different objects. A demodulation object is a logical object inside the The Frontend Engine component that represents different combinations of tuner and demod hardware blocks. The demodulation object can tune to and demodulate all digital and analog broadcast standards, such as DVB-S/T/C, DVB-S2/T2, ITU-T J83 B Annex A & B (cable), etc. A common logical object type is used to exploit similarities between different types of digital and analog RF sources. Different types of source objects can be instantiated with different configuration parameters to support various use cases. The demodulation objects are logical objects that demodulate an RF input signal. Each demodulation object can either perform a digital demodulation or analog demodulation or both, depending on the capability of each of the logical objects.

Transport Engine: The Transport Engine (TE) provides functionality for the processing of transport MPEG streams. Additional variants are provided to sup-

port operator differences. The Transport Engine exposes the functionality through an API that is HW independent, meaning that the API does not expose any HW specific control. All HW specific configurations are centralized and performed at STKPI initialization. A default configuration is provided by the system that should fit the majority of customer needs. However, changes to this default can be made to suit exceptional use cases; a significant number of the APIs in this document are only required to facilitate such exceptional use cases. Thus main task is to do PID filtering of Transport stream via demuxing.

Streaming Engine: The Streaming engine (SE) provides functionality both for the decoding of streams of media data and rendering to a suitable device at the correct time and for encoding media data in real time for transmission of local storage. The functions can be combined in order to enable real time transcoding of media data.

Display Engine: The Display Engine (DE) also called Video Backend (VIBE) is responsible for displaying the video and audio content on HDMI which is given by SE. There are mainly four objects of DE:

Devices: A device is an object owning all of the sibling objects. Most current chipsets have only one single device, but there have been some chipsets like the STi7200 that had two independent display blocks.

Plane: A plane is an object managing a virtual or a real layer on which some content can be displayed. This content can be a single color (in case of a background plane) or a picture (in case of Graphic or Video plane).

Output: An output is an object managing the various types of video outputs present on a chipset (HDMI, YPbPr, CVBS, etc.).

Source: A source is an object representing a source of pictures. It can be a single picture or a succession of pictures. Depending on the hardware constraints, this source can be connected to one or more planes. The source will in fact feed all

the planes that are connected to it.

2.2.2 UserSpace

UserSpace contains those componets which are directly not having privileged permissions like KernelSpace components. These components are as:

PCPD: This is protocol developed by ST for communication between CM and CM master. The main functionality of STPCPD driver is to transfer control command between eCM and eSAFEs. STPCPD driver uses specific packet format to transfer control commands according to PCPDv2 protocol.

eCMMaster: The main functionality of eCMMaster application is send/receive control pcpd packets to/from eCM to guide eCM to start and come online. eCMMaster is responsible for NVM exchange, and software download.

NVM(Non-Voletile Memory): The nvmdb library provides an API interface for NVM access. Whenever the user seeks to retrieve data, the request will be sent to the nvmdb library. nvmdb library manages all the NVM data. NVMdb will first try to retrieve data from the local cache. Local cache is a memory area on the RAM maintained by NVMdb library. It flushes this cache into the flash after a specific time interval. This database include CM related data like MAC address of CM, CM certificate, etc.

eRouter: The erouter is an (eSAFE) device, normally implemented in conjunction with cable modem device. The erouter can also deployed without using the cable modem entity. It can also function as a wireless router and performs the functions of a router and includes the functions of a wireless access point.

GWDevControl: The main functionality of gwdevcontrol application is receive events from eCMMaster application and to take actions when reset event is issued from eCM to eCMMaster. gwdevcontrol directly communicates with eCMMaster.

gst-apps: This module contains a gstreamer application using playbin2. This provides a Media Player, a Zapper and a Live Recorder. The GStreamer core module is needed.

SwDwClient: The main functionality of SW DwClient is to check the software image availability of backend download the image, flash, and re transmit the frontend images.

Chapter 3

Introduction to DOCSIS

Introduction of cable data service to cable television subscribers is one of most successful stories in history of internet. In few years Multiple system operators transformed themselves from being providers of entertainment video into being most popular providers of both entertainment and affordable, broadband data services in North America and Europe. All credit goes to DOCSIS (Data Over Cable Service Interface Specification). The first standard was developed by CableLabs in Mid 90's in order to standardize IP traffic over North America and South America cable networks. This standard was then adapted by European cable plants around 2000. ISTB is EuroDOCSIS compliant so that further explanation is given in context of EuroDOCSIS. Some important milestone in history of EuroDOCSIS is as:

In October 1999, decision was made among group of European cable operators to create European version of DOCSIS specification. The main reason behind that was that DOCSIS specification was the only mature specification available at that time. Four months later they adapted DOCSIS by adding Annex N.

In may 2000, EuroDOCSIS certification board was founded. This board decides if product went through certification process or not. The first certification was started in June 2000. In 2004 EuroCableLabs (ECL) was founded which then turned into

Cable Europe Labs (CEL). In 2014 cable Europe Labs has merged into CableLabs as most European MSOs were now member of CableLabs.

Major goal of EuroDOCSIS is to create means of transporting data bidirectionally across the existing Hybrid Fiber Coax (HFC) plant between subscriber devices and the internet, at high speed. Various versions of EuroDOCSIS are explained as below[10]:

- First version of EuroDOCSIS-1.0 was introduced in 1999 which was basic principle architecture for communication between its two key components CM and CMTS. Enhanced version of this as EuroDOCSIS-1.1 was released in April-1999 after four months with enhanced Quality of Service and security capabilities.
- EuroDOCSIS-2.0 was released in 2001 with enhanced upstream transmission speed using advanced modulation techniques. In this version of EuroDOCSIS, with QAM-64 single channel was able to transmit around 38Mbits/s and for QAM-256 speed was 51Mbits/s.
- EuroDOCSIS-3.0, currently used standard, was released in 2006. Both upstream and downstream channel speed has increased in this version along with support of IPv6.
- In October 2013, latest version 3.1 has been release with addition of OFDM and advance error correction techniques. Commercial deployment of 3.1 will start in 2016.

3.1 Reference Model

Reference model is shown for data over cable architecture is shown in figure-3.1. Layered Architecture. The architecture has two key components: Cable Modem

& Cable Modem Terminating System which are connected to each other via HFC network. The CMTS is connected to cable operator's backbone which links EuroDOCSIS system to the internet. Customers can access internet via their PC, generally referred as Customer Premises Equipment, by connecting it to CM.

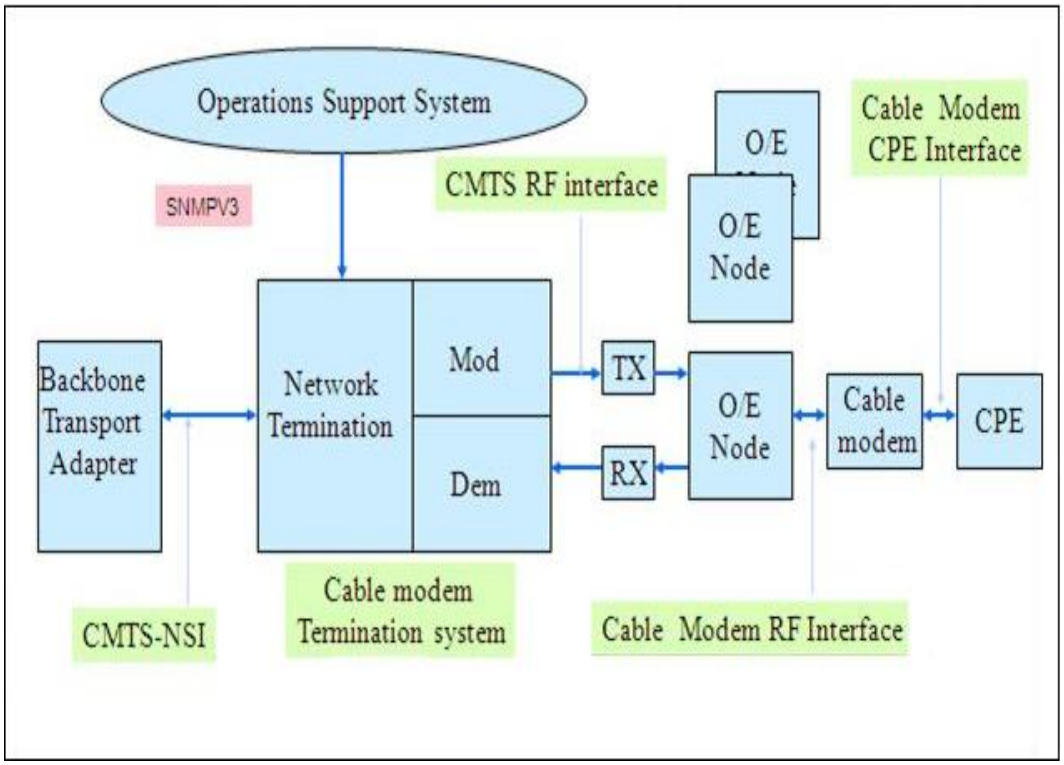


Figure 3.1: Reference Model

CMTS: The CMTS is cable access device which is located at operator's site. Its main task is providing the connection between cable network and the data network (backbone). The CMTS is built out of chassis where one or more linecards reside. A linecard consists of :

- Modulator- to modulate the downstream signal (as many modulators as there are downstream channels)
- Demodulator - to demodulate upstream channel (as many demodulators as there are upstream channels)
- Network termination where data network ends

One CMTS can, dependant from number of downstreams & upstreams, feed few hundred to 10000 homes. There are two types of CMTS exist: bridged CMTS and routed CMTS.

A **bridged CMTS** "bridges" all traffic that goes through it. That means that it does not modify the structure or content of MAC frame. When packet enters a bridged CMTS, the CMTS regenerates the signal and checks physical address of destination and forwards the new copy only to which address belongs. The bridged CMTS has an IP address but this is only used for management purpose.

A **Routed CMTSs** have access to network layer IP address and contains software that enable them to determine which of several possible paths between those addresses is the best for particular transmission. Routed CMTSs will change source and destination MAC addresses of the MAC header.

CM[10]: Cable modem is a specific type of device which is used to provide bi-directional data services to a customer's end using DOCSIS protocol over cable network. It acts as a gateway to home LAN which connects Customer Premises equipment with Cable Modem Terminating System (CMTS) via cable operator network.

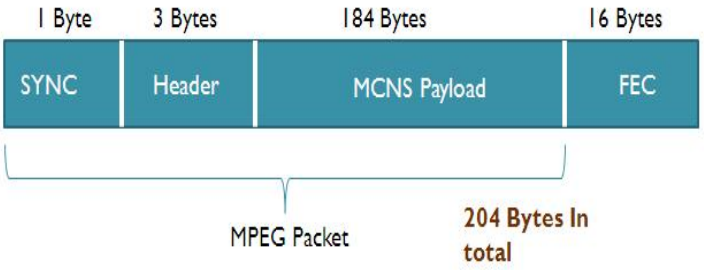


Figure 3.2: Downstream Packet Structure

Downstream: The downstream is continue stream of MPEG-2 packets formatted according to ITU-T H.222.0. The content of MPEG-2 packets is usually digital video, however EuroDOCSIS uses this formatting to its DOCSIS-MAC layer. The real transportation of this stream happens as specified in ITU-T J.83 Annex-A or Annex-B. Annex-A & Annex-B describe both a transmission system that allows for transmission of Low-delay video applications.

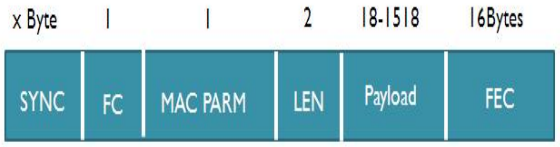


Figure 3.3: Upstream Packet Structure

Upstream: Upstream is output from CM to input to CMTS. Various medium access used for upstream are FDMA, TDMA. In FDMA, the frequency spectrum is divide among different customers, with each customer having exclusive possession of some frequency band. In TDMA, users take turns, each one getting channel bandwidth for little burst of time. Some of parameters are explained in table-3.1.

Table 3.1: Upstream-Downstream Parameters

Parameter	Downstream	Upstream
Frequency Range	108-862 MHz	5-65 Mhz
Modulation	QAM-64 , QAM-256	QPSK or QAM-16
Channel Bandwidth	8MHz	Variable
Symbol Rate	QAM-64 -> 5.056941 msps QAM-256 -> 5.360537 msps	Variable

3.2 Ranging & Registration

When CM is plugged into network, it must establish two way communication with CMTS before it works properly. It is clear that CM & CMTS use two channel to communicate with each other. CMTS uses downstream & CM uses upstream. When CM wants to receive data, it just tune to downstream. To send data CM uses upstream which is more challenging then finding downstream. CM knows nothing meaning centre frequency or bandwidth. It also does not know how far CMTS is which has impact on transmission power. At last but not the least CM cannot transmit whenever it wants to. If two modems transmit at same time, signal will collide.

Once CMTS can communicate with CMTS, it becomes part of IP network. EuroDOCSIS uses IP-based protocols to manage & configure CMs, so every CM must have an IP address.

Communication between CM and CMTS happens at layer-2 of OSI model. When CM Communicate with CMTS successfully, it is said that layer-2 connectivity is established. Layer-2 connectivity is all about gaining access to medium, which in case is HFC network. Communication between CM & CMTS happens through exchange of MAC messages.

CM initialization:

There are different steps that CM has to take in order to become operational and to provide internet to customer. Figure shows detailed initialization process of CM in which first ranging and then registration will make CM operational.

Downstream channel acquisition::

After powered up, CM starts scanning for downstream first. It is searching for valid

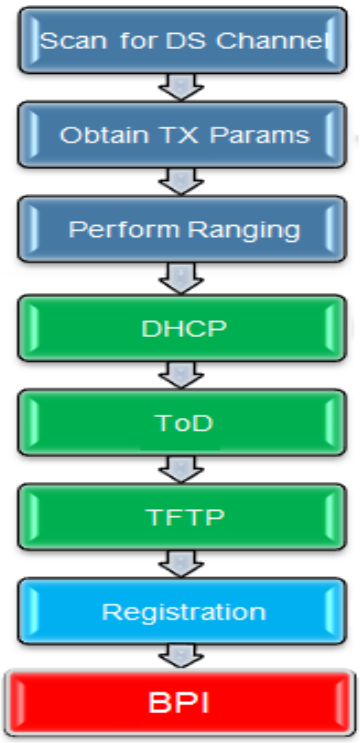


Figure 3.4: CM Ranging and Registration

downstream channel. It may have to search several QAM channels before finding one with EuroDOCSIS data. Possible downstream channels are on interval of 250KHz starting from 112MHz. CM listens to a valid signal when synchronization of QAM symbol timing, synchronization of FEC framing, synchronization of MPEG-2 packetization. Once downstream channel is locked similarly it scans for upstream channel once it collect upstream parameters. A bandwidth is allocated by CMTS after both upstream and downstream channels are locked and this is completion of Ranging. This is also called layer-2 connectivity.

Registration:

After ranging IP connectivity is established between CM and CMTS. Exchange of time of day occurs and CM is given IP address by CMTS using DHCP. This is called registration. This is also called layer-3 connectivity.

Baseline Privacy:

If configured accordingly, modem also has to initiate Baseline Privacy after registration process. Baseline privacy provides cable modem users data privacy across cable network. It does this by encrypting traffic flows between CM & CMTS. If CM is configured to run baseline privacy, CMTS registration is immediately followed by initialization of CM baseline privacy security function. If CM is not to run baseline privacy, the modem is operational after successful completion of ranging and registration. If modem BPI fails, modem is operational (manageable and controllable) but is not allowed to forward traffic from CPE.

Chapter 4

MPEG Streams

4.1 Introduction

MPEG

MPEG is one of the most popular audio/video compression technique. MPEG is not a single standard instead it is collection of standards for different application based on alike concepts and principles. MPEG is abbreviation of Motion Picture Expert Group which was set up by International Standards Organizations (ISO) for compression of video.

Two compression MPEG does:

- Video Compression
- Audio Compression

Both audio and video are multiplexed and transmitted in terms of stream. That is called elementary stream.

An **elementary stream (ES)** is defined as output of video or audio encoder. As its name suggests it is elementary meaning it contains only one type of data, either

audio, video. An elementary stream is commonly calles as "elementary", "audio", "video", "data" streams. The format of ES is as according to the codec or data of stream but it carries a header when packatized into Packatized Elementary stream (PES). A video ES contains all the video data of a sequence with header and sub parts.

In practical purposes, the continuous ES data either audio or video from codes is broken down into packets. The headers in these packets which contains time stamps for synchronizing are identification of the packets. A **Packetized Elementary Stream(PES)** defines carrying of the output of an audio or video encoder (elementary streams) in packets within MPEG program stream and MPEG transport stream.

4.2 Multiplexed MPEG

Following two types of multiplexing is permitted by MPEG

MPEG Program Stream: It is set of densely coupled Packetized Elementary stream (PES) packets on same time-line. These streams are appropriate for transference in error-free environment. This type of data can easily be processed by software. Video playback and network applications are implemented using these streams.

MPEG Transport Stream: PES packets are cracked into fixed sized transport packets which provides technique to combine one or more streams independent of time bases. Thus it can be used in transmission where probability of packet loss is more and even if packet loss is tolerated. Advantage of using transport stream is that it is possible to send more than one program at a time.

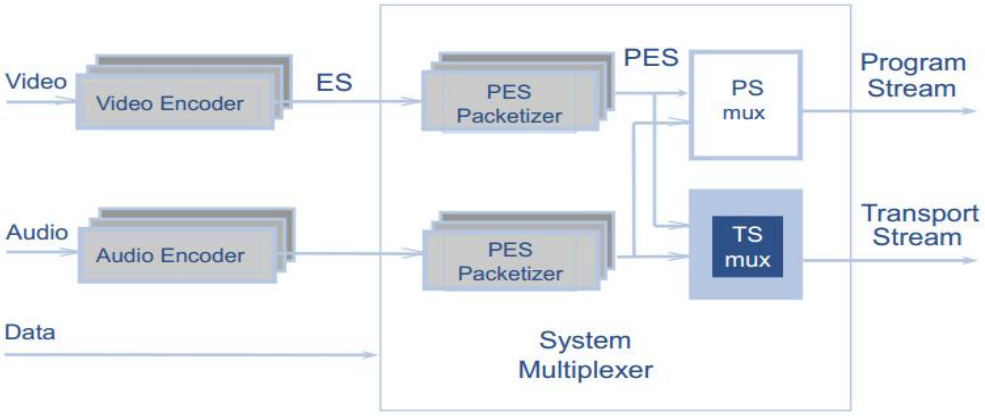


Figure 4.1: MPEG Streams[3]

4.2.1 MPEG Transport Streams

A transport stream contains a sequence or series of transport packets of fixed size of 188 bytes. Each packet is made up of 184 bytes of payload and 4 byte header. 4 byte header is divided into 13 bit of packet identifier (PID), 1 bit of Transport Error Indicator, 1 bit of Payload Unit Start Indicator and 1 bit of Transport Priority. This is shown below in figure-4.2.

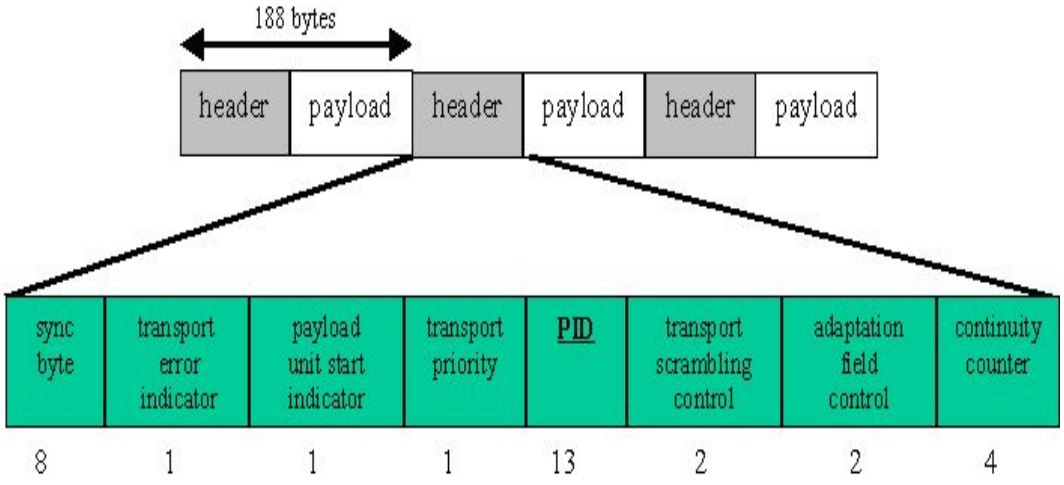


Figure 4.2: Transport Stream Packet Format[3]

The transport stream packet format is illustrated in figure-4.2. Figure-4.3 shows two elementary streams are multiplexed in one MPEG-2 transport. Each packet associated with PES has unique PID value in packet header (64 and 51 in the figure-4.3). Video packets have PID 51 and audio packetes have PID 51. As it is seen that there are more video content/data than audio and they are not evenly spaced oin time. Thus MPEG-TS is not Time division multiplexing. Packets having any PIDs can be placed into TS at any time. If no packets are available then the multiplexer inserts null packed which are identified with PID value of 0x1FFF.

Figure 4.3: Transport Stream Format[3]

4.2.2 Types of the MPEG-TS

MPEG-TS may be used widely as in Digital Video Broadcasting (DVB) as well as in communication network. A strong error correction mechanism makes MPEG frames so robust. It is constructed so close to characteristics of cable or radio channel that it expects Bit Error Rate (BER) of better than 10^{-10}. All variants of DVB has its own coding defined and modulation techniques for particular channel environment.

4.2.3 Single and Multiple Program Transport Streams

A Single Program Transport Stream may be referred as only one TV program or stream which contains a video and appropriate video.

Information necessary to decompress and regenerate the encoded stream is also stored in SPTS. It may also have other type of PES other than just audio and

video. A common timebase is shared by each PES.

In practical use in DVB, more than one SPTS streams are muxed together to generate Multiple Program Transport Stream. This aggregation also carries program specific control information required for co-ordinfation.

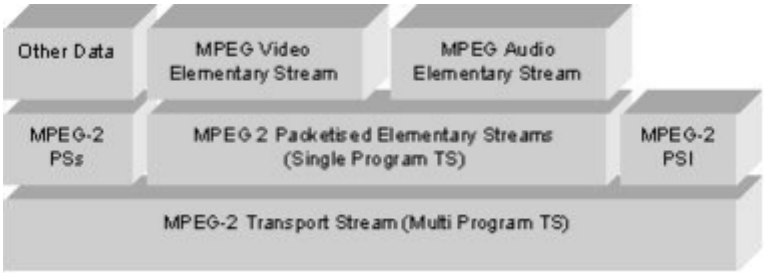


Figure 4.4: Multi Program Transport Stream[3]

Transport stream consists of so many related elementary streams (audio and video streams). These elementary stream must be decoded in synchronisation to make sure audio-video playback is as expected. In case of digital TV program or radio program, it is necessary for stream decoding to be synchronised, while in case of programs offering downloading, it is not necessary. Time stamps are attached to control synchronisation in transport stream.

Chapter 5

Version Control System

A version control system (or revision control system) is a system that tracks incremental versions of development file-system including files and directories over the course time. Obviously, solely stalking the different versions of a user's files and directories isn't very captivating in itself. Permitting to explore changes which outcame in each version is what makes version control system effective.

A Revision Control System is repository, collection of stored file, usually source code of computer programs with observed access. All changes made to source code is stalked with who made the change, time stamps of change, rational for change and enhancement introdecued or previous problem resolved by the change.

Such systems are significant for distributive, participative or colloborative development. For a large software development project, potential to stalk all the changes with time stamps and to revert back the changes if necessary, can make all difference between well-controlled and managed system and uncontrolled first come, first served system.

5.1 Repo Tool

Repo is repository management tool which is build above Git. Repo combines together many Git Repositories and able to upload to revision control system and software developments flow becomes automated. It is not designed to take place of Git, it enhance the capabilities of Git and makes it easier to use. It is an executable Python script which can be stored anywhere in the system path. In working with source file, repo is used for across-network operations. For e.g. by executing only single Repo command, it is possible to download multiple repositories into local working directory. Following command is generic for Repo usage:

```
repo COMMAND OPTIONS
```

Brackets [ ] contains optional arguments. Information of any command can be obtained when repo is installed by running

```
repo help COMMAND
```

Repo is installed by giving following command. It gets installed in current directory. A .repo/ directory is created which has Git repository of Repo source and some standard manifest files. Very important file called manifest.xml is generated which is symlink to selected manifest in .repo/manifests/ directory.

```
repo init -u URL [OPTIONS]
```

Options:

- -u: URL is given from where manifest directory to be retrieved
- -b: Specify manifest-branch.
- -m: Manifest file in directory is chose. By default it chooses default.xml, in case

of no manifest.xml.

repo sync [PROJECT_LIST]

New changes are downloaded and local environment is updated. If it is supplied no arguments then all project files are synchronized.

5.2 Git

Git is an open-source version-control system developed to maintain very large projects those are distributed over multiple repositories. Local operations such as local commits, branching, edits and diffs are permitted by Git. As mentioned in section-5.1, Repo tool is applied for across-network operations. For example, Either using Git commands for each component or using one repo command files from multiple repositories can be downloaded. This gives freedom to install components at desired locations.

The considerable difference between Git and any other VCS is the manner in which Git deals with data. Mostly other systems takes care of the list of files which encountered change. The fashion in which information stored here is they possess bunch of files and changes occurred in those files over the span of time.[7], as shown in figure-5.1.

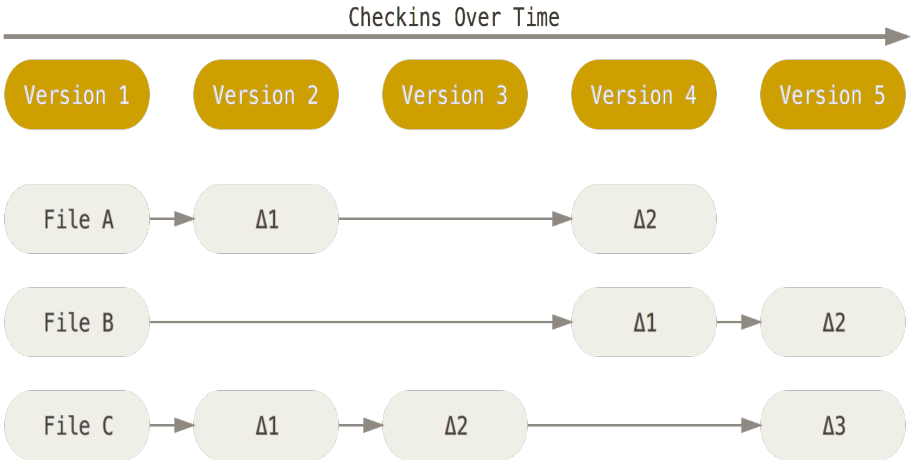


Figure 5.1: Stored as change in each file[7]

Git doesn't take care of the data in this manner. It stores the data in fashion of pictures or snapshots of a mini file-system. Every time there is a commit in Git meaning some change in files, a snapshot of what all files look like at that moment is taken and stored as a reference to that snapshot. To be well organized, if any of files have not changed, it doesn't store the file again and it just a link them to the previous identical files. Git takes care of data like **stream of snapshots**. Figure-5.2 explains this graphycally.

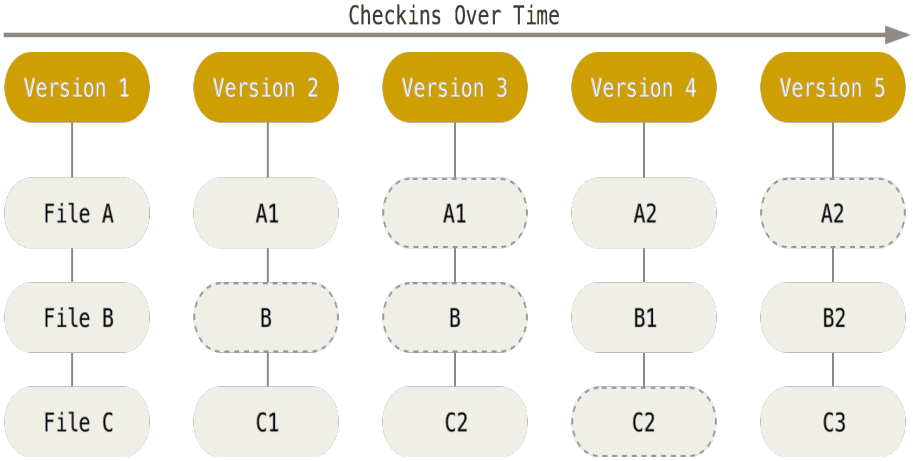


Figure 5.2: Snapshots over the time are stored[7]

The Three States

Git has three main states in that project files can reside in: committed, modified, and staged. Committed state means that the data is stored in local database. Modified state means that file is modifies but have not committed it to database. Staged state means that modified file is marked in its current version to appear in next commit snapshot. This takes to the three main components of a Git project: working directory, Git directory and staged directory.

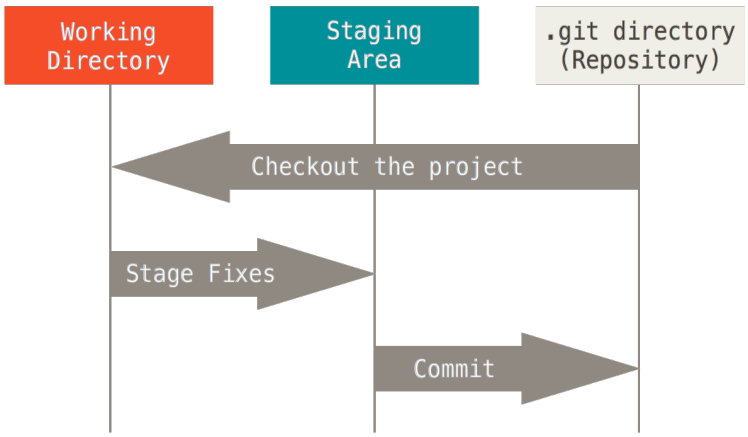


Figure 5.3: The Three States[7]

The Git directory is the direcoty where metadata of the project and object database is stored. This is the most essential part of Git. When user does git clone this file directory is copied. The working directory is a local database of one version of project which is ready to be used or modified. These files are pulled out of the database in the Git directory. The staging area is intermediary area inside Git directory which has information of what will go into next commit. Its sometimes referred to as the index, but its also common to refer to it as the staging area. The Git work flow is as follows:

1. Update files with changes needed.
2. Add those changes into staged area and take snapshot.

3. Commit those changes which are already there in staged area and save the changes as snapshot permanently.

Git Commands:

◇ **git clone**: It copies the git repository to current working directory.
◇ **git add**: It adds locally changed files into staging area. ◇ **git rm**: It removes the file from staging area and local directory.
◇ **git commit**: Once changes are done and added to staged area, this command will commit the staged area content to project history.
◇ **git status**: This command shows the status of staged area with full details. Three types of files are shown over here, untracked which are unchanged, modified which are changed but not yet added in staged area and staged which are added to staged area.
◇ **git checkout**: It is multiple used command meaning it checks out a different branch or it can be use to switch to other branch by updating index.
◇ **git reset**: Current index is reset and reverted back to last commit state.
◇ **git diff**: If gives the difference between current index and current directory, if there is any.
◇ **gitk**: Graphical Tcl/Tk based interface to a local Git repository.

Chapter 6

Testing

In general, testing is finding out how well something works. In hardware and software development, testing is part of the overall process to check whether objectives are being met or not. For e.g. in software development, project objectives are tested first and then design is prepared. When design gets completed, coding is done and at various points code is tested at module level by programmer, by group of programmer at component level and in the end all components are combined together at system level.

Software testing is activity of assesment of s/w product to identify differences between given input and expected output. It also evaluates quality of software product. This process must be done in simultaneous with development process. Thus software testing starts with verification and ends with verification.

Verification: Verification is the process to make sure the product fulfills the predecided objectives at starting of development. In other words, to make sure the product behaves the way we want it is designed to.

Validation: Validation is process of being assured that the product is confirming specified requirements in the end of developement process. In either words, to chek whether customer's requirements are met or not.

6.1 Software Testing Methods

There are different methods which can be use for Software testing as follows:

6.1.1 Black Box Testing

In this technique of testing, interior working knowledge is known by tester. Tester is clueless about system architecture and source code. Typically in black box testing, a tester will interact with system's user interface and will examine output by providing inputs without knowing how system works.

6.1.2 White Box Testing

White box testing also called open box or glass testing in which detailed investigation of structure of code and internal logic is done.In order to perform white box tesing, a tester has to have full knowledge of source code and system architecture.

6.1.3 Grey Box Testing

Grey Box testing falls in between black box and white box testing where tester has limited knowledge of internal working of system. Unlike black box testing here tester has access to some design documents and has some knowledge of source code. Thus tester have more ability in testing and can prepare more test scenarios.

6.2 Types of Testing

Types of tesing are immense in numbers. Some of them are[5]:

- Unit testing
- Functional testing
- System testing
- Stress testing
- Performance testing
- Regression testing

Unit testing: Unit testing is process of testing each and every unit or related bunch of units of system. It comes under white box testing. Here usually tester checks that system output for given input is same as expected.

Functional testing: Functional testing checks that specified customer requirement functionality is working properly. It comes under black box testing.

Stress testing: This is type of testing which tests the system performance under adverse situations. Testing is done beyond limits of the specifications. It comes under black box testing.

Performance testing:This testing is conducted to evaluate the speed and potential of system to assure that system is producing results withing specified time as mentioned in performace requirements. It comes under black box testing.

Regression testing: Regression testing is the testing after modification of a component, group of components or system to be sure that modifications are working

correctly and not affecting to other modules to produce unexpected results. It comes under black box testing.

Of the above discussed testing types, following tests are included for Interactive Set-top Box testing:

- Functional testing
- Parallel testing
- Stress testing
- Robustness testing

6.3 Sanity testing & Smoke testing

Smoke testing

Initial testing process which is conducted to check whether the product under test is stable/ready for further testing is called Smoke testing. This term comes from hardware testing as in hardware testing initially it is checked that it does not catch fire or smoked when powered on.

Few test cases are required to be created once to perform testing before starting Smoke testing. These test cases are then executed in advance to start actual testing to check **critical functionality of the program is working fine or not**.

Sanity testing

Once a Software build is received with minor issue fixes in code, Sanity testing is carried out to ensure whether the bugs reported in previous build are resolved and no regression is introduced because of these fixes i.e. previous functionalities are working fine. The main purpose of Sanity testing to verify the planned functionality is working as expected. In figure-6.1 graphical representation of Smoke testing &

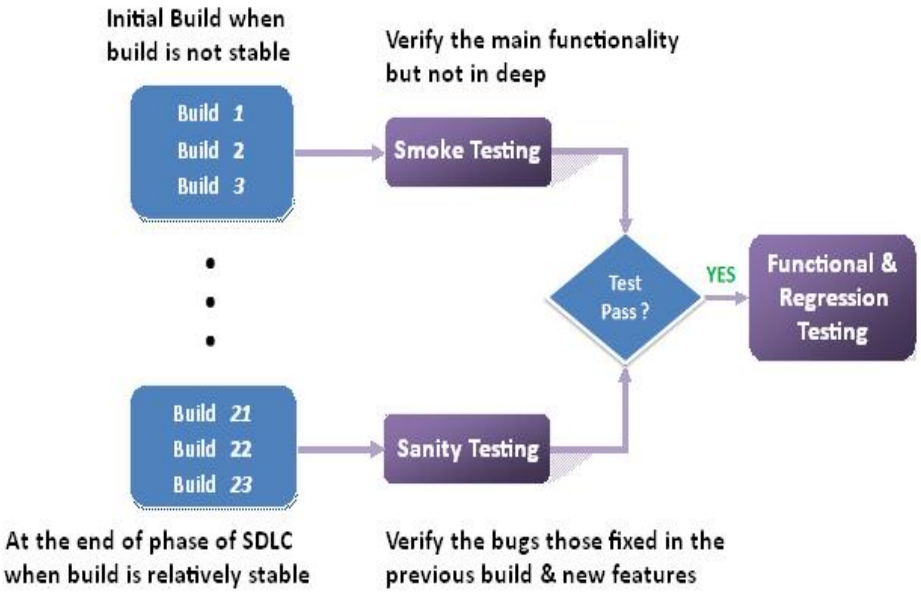


Figure 6.1: Smoke v/s Sanity testing[6]

Sanity testing in software testing is illustrated. Here are the few consolidated points of Sanity testing:

- Sanity testing follows narrow and deep approach with detailed testing of some limited features.
- Sanity testing is used to verify the requirements of end users are meeting or not.
- Sanity testing to check the after minor fixes the small section of code or functionality is working as expected & not breaking related functionality.

6.4 Test Case

Test cases are the collection of various steps, conditions and inputs which are used to perform the testing. The main object of this activity is to ensure whether the

Software Passes or Fails in terms of its functionality and other aspects. Moreover test cases are written to keep track of testing coverage of Software. In general there is no standard template to write test case.
Following test cases are mainly designed for ISTB testing purpose:

- Linear TV

  This use case checks basic functionality of set-top Box and checks 8-tuners of STB in which 8 different channels are locked on 8-different tuners one by one basis. This is the most basic use case with respect to ISTB validation.

- ZAP

  Zapping implies changing channel while one is already going.

- Trick Mode

  Increasing the speed of playback in forward & backward direction

- DVR

  To store video in digital format into local storage

- MediaPlayback

  Play Recorded Media

- Data Browsing (Wired & Wireless)

  Internet browsing using Ethernet & Wi-Fi

- Robustness

  To check ability of system to cop-up with run time errors

6.5 Patch

A patch is a piece of software designed to fix problems with, or its supporting data. It involves fixing previously reported bugs or security vulnerabilities and improve performance. Although patches are created to fix problems, poorly designed patches

sometimes can introduce new problems. Patch is a UNIX program that updates text files according to instructions contained in a separate file, called a patch file. The patch file (shortly patch) is a text file that consists of a list of differences between original and updated file and is produced by running the related diff program. Updating files with patch is often referred to as applying the patch or simply patching the files. Updating files with patch is often referred to as **applying the patch or simply patching the files**.

First the contributor creates patch and submits to maintainer. The project maintainer then studies the patch and applies to main code. Various tools are available for that. These tools make patch creation and management easy and efficient. These tools are key in building an active community of contributors to open source project development.

Applying the patch:

Patches can be applied using following two ways:

• **patch -dry-run -p1 patchname**

It doesn't actually change the files. It checks what would happen if patch was applied.

• **patch -p1 -I patchname** : This is to apply patch.

• **git apply path/file.patch** : This is another way to apply way using Git.

This -p option strips out prefixes according to the option given with it. If patch was generated using git diff then -p1 would be appropriate.

Chapter 7

Linux and Hardware Environment Setup

7.1 Linux Environment Setup

Any flavour of linux either fedora or ubuntu is required as software environment for testing purpose.

7.1.1 ssh Key Generation

Secure Shell (SSH) is a cryptographic network protocol for secure data communication, remote services and other secure network services between two networked computers that connects. Connection is done between a server and a client via a secure channel over an insecure network. Public and private keys are generated by ssh-keygen. ssh-copy-id copies local host's, client's public key to remote host's, server's authorized key files. It also assigns proper permissions to remote-host home, server home, /ssh; and /ssh/unauthorized keys.[1] Omit following steps if access to git repository is already granted. Use following commands:
ssh-keygen -t rsa -C
emailaddress : f /.ssh/id rsa

7.1.2 Build Process

he term build refers either to the process of converting source code files into standalone software artifact(s) that can be run on a computer. One of the most important steps of a software build is the compilation process where source code files are converted into executable code. In software version, the build number is often used as a version identifier. As software is divided into various components/modules, all components, kernel, it is necessary to have Makefile for each.[1]

1 Source files are converted into object files by compiler.

2 These object files are then converted into Executables by Linker.

Go to /build path/build/sdk2-build.b2163-d127_a9/ as a normal user and follow the below commands.

- make clean: To clean previously compiled build
- make modules: All sdk2 modules are built
- make module_name: Specific sdk2 module is built
- make .clean_module_name: Specific module is cleaned
- make .modules_install_module_name: Specific component is installed
- make all: To build and install kernel, all sdk modules

7.1.3 Set IP Address to download the object

Executable generated, as explained above, can be run on the target platform using JTAG connection and logs can be observed on serial port while execution of source code on the target. For that Micro Connect is connected as interfacing equipment between target and host. So IP addresses are assigned to Micro connect and target board. Go to build path/build/config.in Set the JEI, TARGET IP and Gateway IP.

- JEI = The IP address of Micro connect.
- TARGETIP = IP address of the target board where kernel loaded.
- GWIP = IP address of Gateway.

7.1.4 Test Execution

Once build process is completed and executable files are generated related to source code compiled, to test ISTB devices and drivers, this target image is to be installed on the development board. Once downloaded into target board, to perform validation process various use cases are performed. These use cases are in actual requirements by the customer. To validate all use cases board is first booted. In order to do this, configuration file in the build directory is updated for JEI, TARGETIP & GATEWAY IP.

- make run

The board is booted after execution of this command. VMLinux which is executable file of kernel and contains some other object files, is loaded into the target board SoC.

Booting is the first process when target board is initialised. All the hardware components are initialised as well as they are brought together to work and load operating system which make the system operational. As soon as board is powered up, the control is given to bootstrap process in BIOS in ROM. This process is responsible for initialisation of hardware.

- ssh rootIP_address_of_target

Log in as a root user on the target platform. This should be done on different terminal as done to boot kernel.

- ./framework go.sh

Load Modules: Various modules developed by ST which are required to run use cases are loaded. Linux utility “Modprobe” is used to accomplish this. It loads run time modules and its dependencies.

- Run/Application

7.1.5 SDK2 Environment

SDK2 provides a unified software devlopment platform for STs class 4 devices using the Linux operating system.It includes low level drivers (STKPI) and an application layer (STMediaFramework).[1]

- STMediaFramework

User level set of managed APIs and components for multimedia applications devlopement.

- STKPI

Kernel level managed API.

7.2 Hardware Setup

To set up the hardware to carry out the testing following are the required components. Hardware requirements as follows:

- STMC
- Target Board
- Stream Server
- JTAG
- UART
- HDMI cable for connection with TV
- TV
- Power adapters

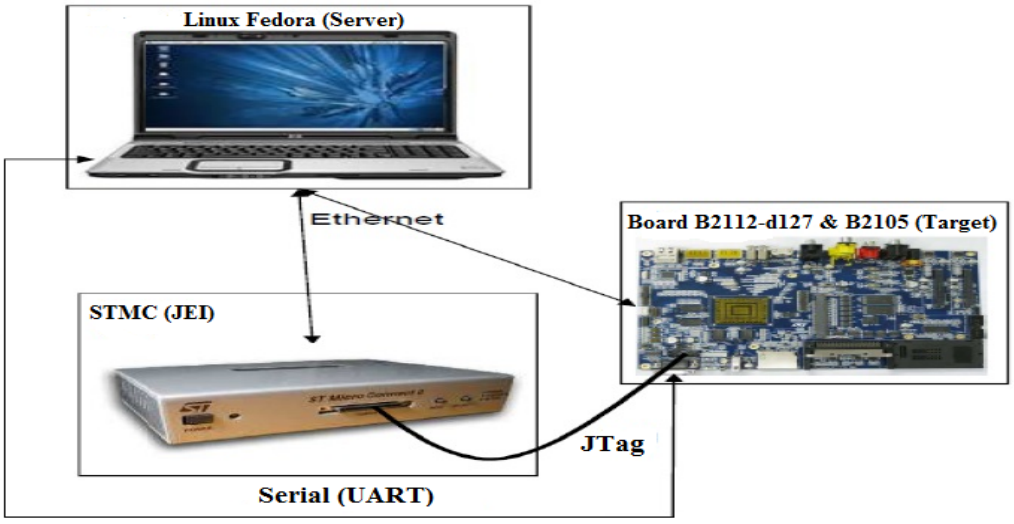


Figure 7.1: Hardware Setup[1]

Different components are:

SoC:

The System on chip contain two ARM processors, one SH-4 (used for floating point calculations), One ARMv7.

ST Microconnect:

MicroConnect is used to load compiled image of kernel into targe SoC. ST Micro connect(STMC) is a host-target interface from STMicroelectronics. Target development board board's JTAG connector is connected with STMC, image is downloaded and programs are debugged. It is easy to install and use the ST Micro Connection Package provides software utilities and firmware,including Target Packs for certain ST evaluation boards.

Stream Server:

All the required testing video steams are stored in stream server to be viewed in digital format, during testing of video playback. In satellite feed, tuner receives via dish antenna. Here in this case of cable feed, streams are supplied via stream modulators. A particular stream has its data rate, symbol rate, FEC, etc. To perform various tests on different channels and symbol rate stream server is required. Modulator cards are connected with stream servers. These modulators are connected to RF tuner of target board via RF cable. Modulator card converts into RF frequency and sends it to RF tuner card using RF cable.

JTAG:

This intermediary medium gives ease to load kernel image into development board.

UART:

Debug information is obtained via serial communication between target board and host machine using UART.

Chapter 8

Software System Integration & Validation

8.1 Software Integration

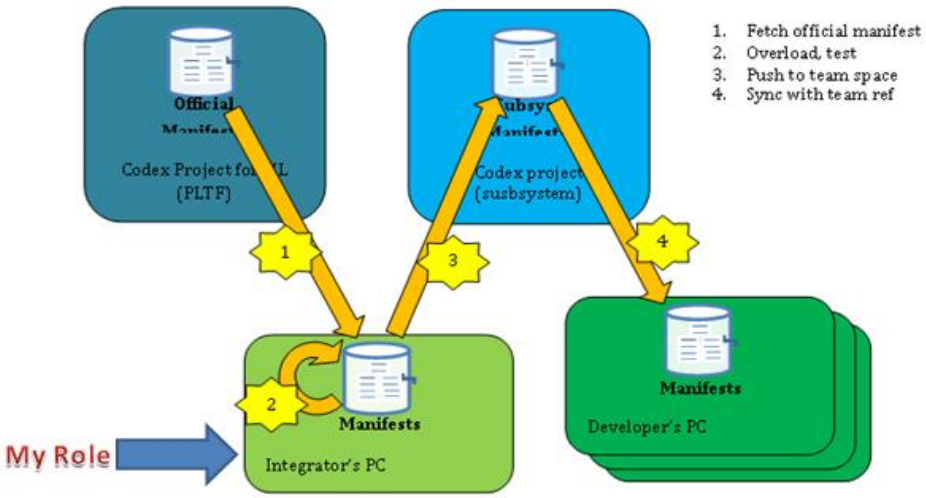


Figure 8.1: Software Development process

Software system integration is task of binding together individual tested software components into one integrated whole. A software is said to be integrated when

subsystems are divided into components or product is divided into subsystems. The component or subsystem are tested individually and then integrated together to make whole software. Software system integration either looks like a discrete step towards end of the development cycle of software development process. **ISTB Software Integration** is continuous intefration which is much less risky approach where components are integrated as and when they are developed, also called unified methodology. Figure-8.1 shows the software development process highlighting integration process.

As seen in section-2.2, ISTB software is divided into KernelSpace and UserSpace, each having various components, it is integrated using repo and git tool by including various components in menifest.xml file.

8.2 Software System Validation

Software system validation is process of determining whether the system complies with requirements and all intended functionalities are perfectly performed and it meets customer's needs & goals. Validation is done at the end of development process and is next level of testing after verification. Verification is done in starting of the development with the question "Are we building the product right?", while validation is done at the end with question "Are we building the right product?".

Various ISTB use cases are discussed in section-6.4. **ISTB validation** also known as **System Testing** includes:

- **ZAP**, in which following use cases are performed (41 use cases in total):
  - CODEC tests
  - Resolution tests like 720p, 1080p
  - QAM testing
  - Stress testing also called overnight testing
- **MediaPlayer**, in which Locally stored Media is played with different codecs. Along with that Forward & Backward trickmode testing is also done. In addition to that JPEG image testing with their rotation is tested. (10 use cases in total)
- **Digital Video Recording**, where live channel is recorded on local storage increasing progressively from tuner 1 to 8 parallely. These use cases include Live A/V decode along with background recording. (26 use cases in total)
- **Data Browsing**, includes browsing internet on wired and wireless connection. This use case is performed parallely with Linear TV use case which is full functionality of ISTB. (5 use cases in total)
- **Robustness**, (9 use cases in total)

8.3 Prerequisite

SDK2 must be installed and nfs services must be started on linux machine. Entries must be added for nfs-mount in /etc/exports file. Sanity can be performed manually and in automated manner. Python is required for automated execution of sanity. First it is explained in manual manner and then in automated environment.

Channel Configuration[1]:

Add channel configurations in .conf file as follows:

ChannelName : Frequency : Inversion: SymboleRate :FEC : Modulation:
VPID : APID : PCDPID : VTYPE : ATYPE : TTXPIDF : TTXTYPE
Where:

• Channel Name: Name of channel

• Frequency: freq. at which channel is broadcasting

• Symbole Rate: symbol rate is the number of symbol changes (waveform changes or signalling events) made to the transmission medium per second using a digitally modulated signal or a line code.

• FEC: Forward Error Correction (FEC) or channel coding[1] is a technique used for controlling errors in data transmission over unreliable or noisy communication channels.

• Modulation: Modulation technique whether QAM-64 or QAM-256

• VPID: Video Program ID

• APID: Audio Program ID

• VTYPE: Video type(video codec)

• ATYPE: Audio type(audio codec)

• TTXPID: Subtitle PID

Example:

Channel1 : 663000000 : INVERSION_AUTO:5056941 : FEC_AUTO:QAM_64 :
2318 : 2320 : 2318 : 8 : 3 : NONE : NONE

8.4 Manual Execution

Following are steps to be performed sequentially in order to run various use cases on ISTB :

STEP-1: repo init to current tag and repo sync.

STEP-2: Apply the patch and repo sync.

STEP-3: Go to build directory build the code using make command.

STEP-4: Export environmental variables.

STEP-5: Boot the board using *make run* command.

STEP-6: Follow steps in section 7.1.4.

STEP-7: Configuration entries of channels on .conf file.

STEP-8: Run gst-apps command on target.

Linear TV:

Basic use case of ISTB when Live A/V decode is performed. To test various tuners, option -N0...-N7 is given.

```
root@b2163-h410_a9:~# gst-apps dvb://robots
**************************
GST-APPS version : v2.5.0
**************************
*****************
GStreamer 1.2.2
*****************
Playing file dvb://robots
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
q
Stopping the pipeline...
```

Figure 8.2: Linear TV use case[1]

ZAP:

Zapping in simple words is changing channel while one channel is going on.

STEP-9: Press "d-channel down" and "u- channel up".

```
root@b2163-h410_a9:~# gst-apps dvb://robots
**************************
GST-APPS version : v2.5.0
**************************
*****************
GStreamer 1.2.2
*****************
Playing file dvb://robots
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
d
Trying to lock tuner...
Tuner locked
Playing CAB_MUX channel
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
u
Trying to lock tuner...
Tuner locked
Playing robots channel
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
q
Stopping the pipeline...
```

Figure 8.3: ZAP use case[1]

In case of ISTB Sanity testing, two Linear tv use cases are run simultaneously, one Main and one PIP(Picture in Picture). Zapping is done in Main AV(Audio/Video) decode.

MediaPlayer:

Local stored media is played using gst-apps command as below:

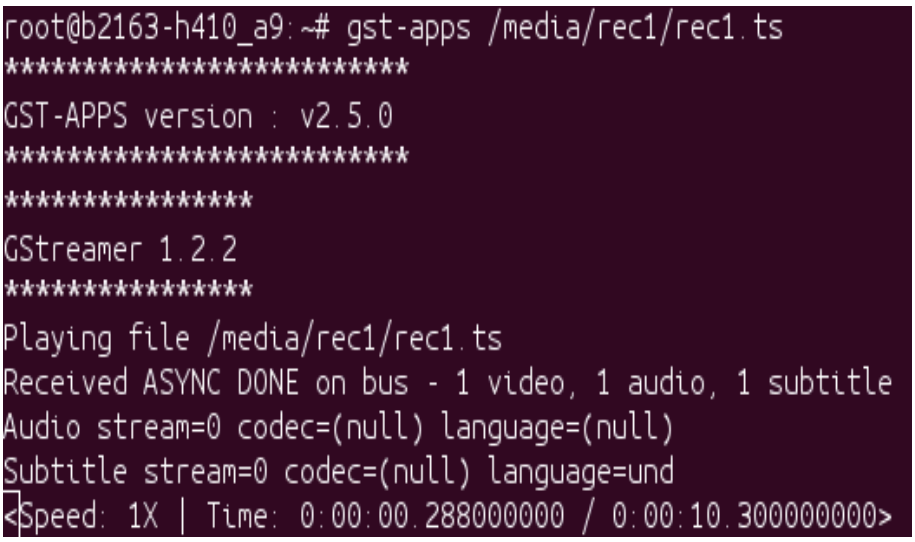


Figure 8.4: Local Media PlayBack use case[1]

Digital Video Recording (DVR):

DVR can be performed in two ways. In first, a channel is recorded while A/V decode is going on. While in second case, channel is recorded in background. As ISTB supports 8-tuners, 2-A/V decode in addition with 8 channel record is maximum utilization of DVR.

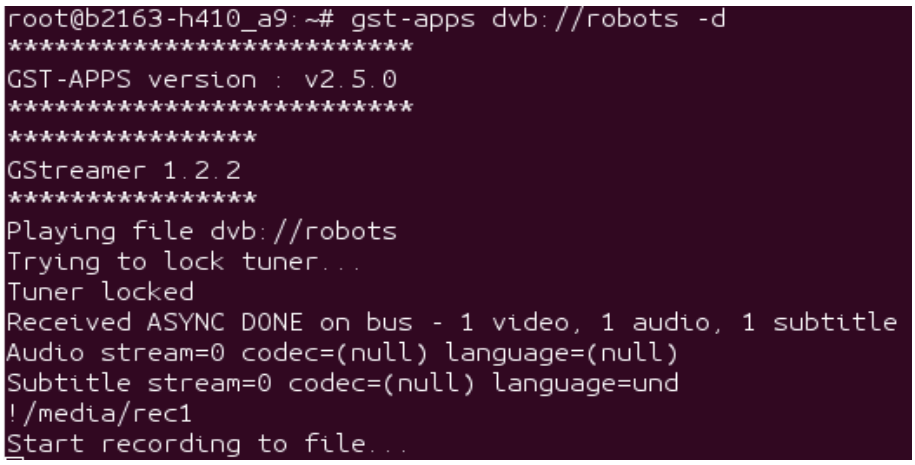


Figure 8.5: Digital Video Recording[1]

Data Browsing:

This use case is related to Gateway part of ISTB. This use case requires cable modem to be operational. Once cable modem is operational, run erouter application on gateway to enable data browsing on Ethernet port as well as on Wi-Fi.

8.5 Automation of System testing

As discussed above, full software system validation includes vast number of use cases and its cumbersome process if performed manually. For automation of testing ST has developed a TestFramework. TestFramework is implemented using several xml and python files. Use cases are written in xml files which calls python files in which execution code of test cases. TestFramework file system architecture and working is explained as:

1. Stream Descriptions:

• This xml contains file names, containers, duration, codex for all tracks, resolutions, rates, stream ids.

• First parses directory tree and calls mediainfo for deep description of multimedia files.

2. Stream locations:

•Aim of automation is to make an independent way of testing where target board can access files from anywhere. Stream locations.xml fulfils this purpose by telling board the location of file depending on mode (http, local).

3.root.xml:

• It contains information about the other xml data files which the framework requires to run the test. It also contains the use-cases to be run, the application used to perform the use-case.

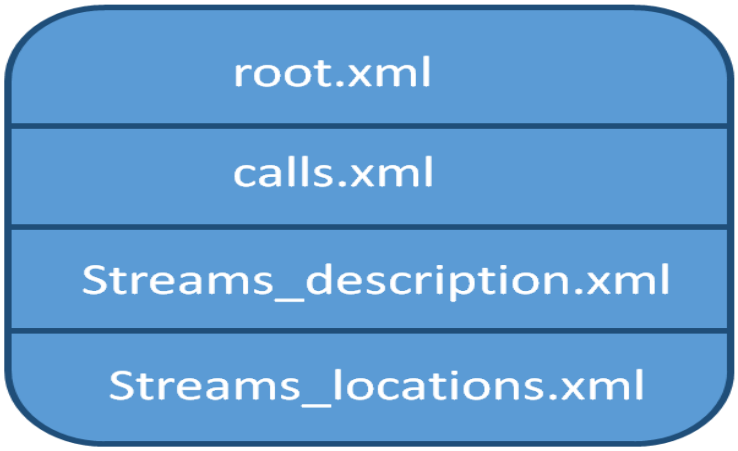


Figure 8.6: TestFramework files[1]

4. Calls.xml:

• This File Contains use case definition. Call to python files goes from this file along with required parameters.

5. use_case.py:

• It is python file which actually contains the definition of python function for particular use case.

Execution via TestFramework

We can launch Test Framework before or after the target has booted.

COMMAND LINE OPTIONS

• General syntax:

main.py [session options] root.xml[root name]*[additional path]*[targetIP]

• The [session option] can be:

1. -initialboot

Reboots the board once before launching the tests

2. -initialbootonly

Reboots the board from the framework and exits

• [root name] is the optional name we can encounter within a launch description of the root file. **To run particular testing, appropriate root_file is given as**

input.

• [additional path] is here for extending the PATH search of the python and XML files within the Framework.

• [targetIP] must be an ipv4 address.

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: ssh the target as root

STEP-4: Load the modules like media player, HDMI Manager etc

STEP-5: Change the directory to mainline manifests test framework directory.

STEP-6: Export some of the environmental variables in run_test.sh file.

Variables are:

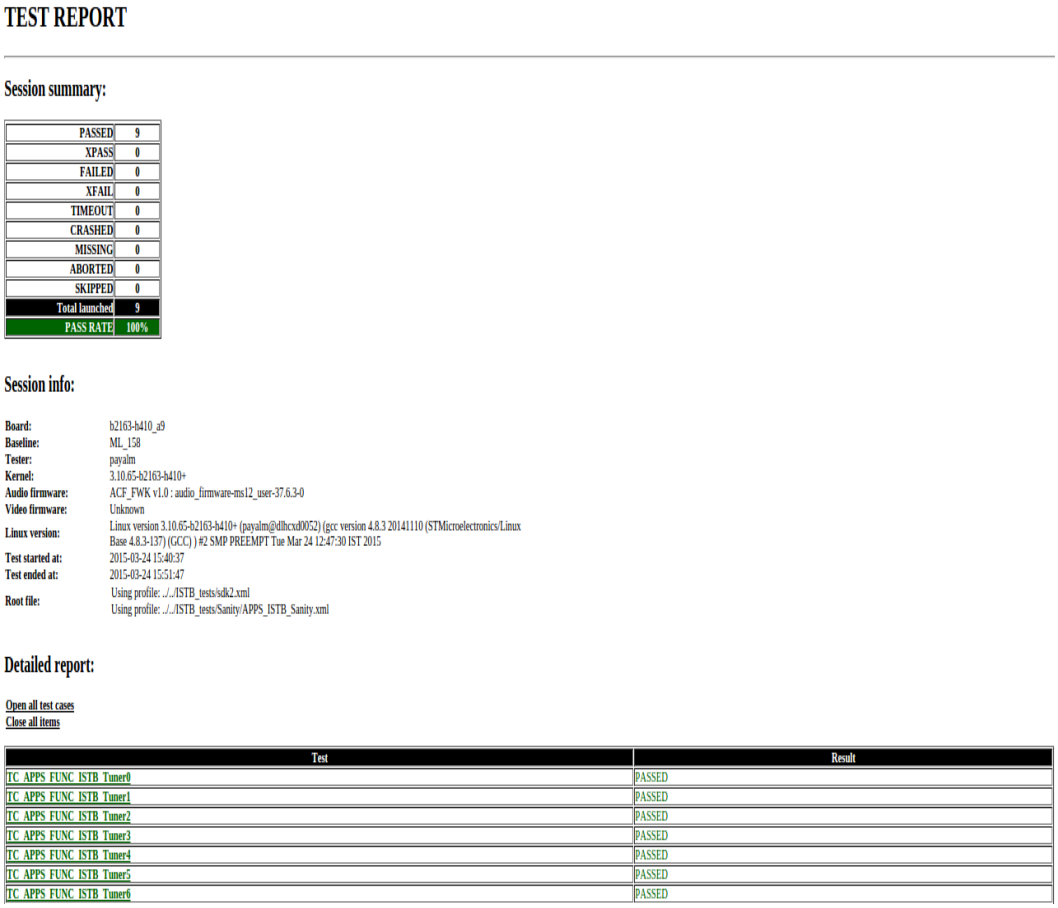
- SERVERIP : IP of host machine
- TARGETIP : IP of the board
- JEI : STMC (ST Micro Connect) IP
- GWIP : Gateway IP
- NFS-SERVER : Network File System IP

STEP-7: Source the above mentioned file

STEP-8: Execute the script main.py along with required arguments to start the tests.

STEP-9: See the Reports in RESULT folder of that particular test framework.

Figure-8.7 shows result generated for sanity using automation.



TEST REPORT

Session summary:

PASSED	9
XPASS	0
FAILED	0
XFAIL	0
TIMEOUT	0
CRASHED	0
MISSING	0
ABORTED	0
SKIPPED	0
Total launched	9
PASS RATE	100%

Session info:

Board:	b2163-b410_a9
Baseline:	ML_158
Tester:	payalm
Kernel:	3.10.65-b2163-b410+
Audio firmware:	ACF_FWK v1.0 : audio_firmware-ms12_user-37.6.3-0
Video firmware:	Unknown
Linux version:	Linux version 3.10.65-b2163-b410+ (payalm@dlhcxd0052) (gcc version 4.8.3 20141110 (STMicroelectronics/Linux Base 4.8.3-137) (GCC)) #2 SMP PREEMPT Tue Mar 24 12:47:30 IST 2015
Test started at:	2015-03-24 15:40:37
Test ended at:	2015-03-24 15:51:47
Root file:	Using profile: ../../ISTB_tests/sdk2.xml Using profile: ../../ISTB_tests/Sanity/APPS_ISTB_Sanity.xml

Detailed report:

Open all test cases
Close all items

Test	Result
TC_APPS_FUNC_ISTB_Tuner0	PASSED
TC_APPS_FUNC_ISTB_Tuner1	PASSED
TC_APPS_FUNC_ISTB_Tuner2	PASSED
TC_APPS_FUNC_ISTB_Tuner3	PASSED
TC_APPS_FUNC_ISTB_Tuner4	PASSED
TC_APPS_FUNC_ISTB_Tuner5	PASSED
TC_APPS_FUNC_ISTB_Tuner6	PASSED

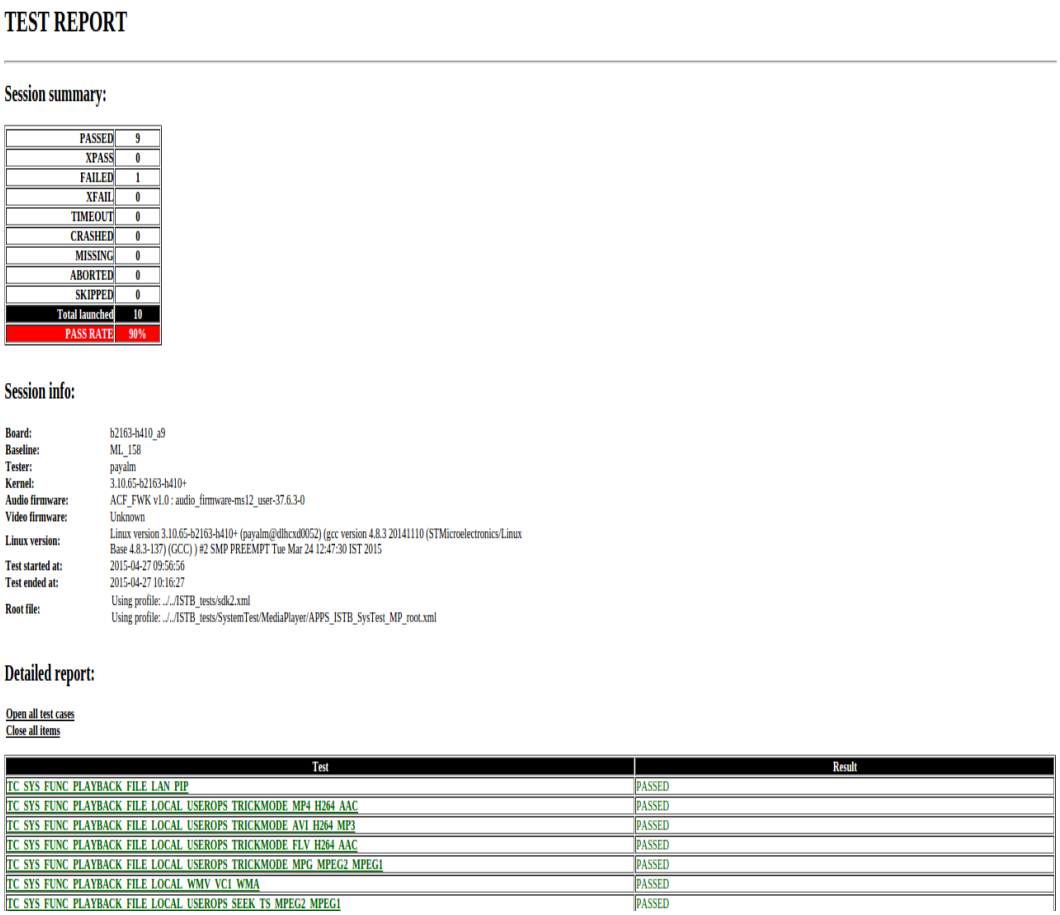
Figure 8.7: Results for Sanity testing

Table 8.1: Verdict Description[1]

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.
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

Table-8.1 gives description of various results of test application.

Figure-8.8 shows result generated for MediaPlayer using automation.



TEST REPORT

Session summary:

PASSED	9
XPASS	0
FAILED	1
XFAIL	0
TIMEOUT	0
CRASHED	0
MISSING	0
ABORTED	0
SKIPPED	0
Total launched	10
PASS RATE	90%

Session info:

Board:	b2163-b410_a9
Baseline:	ML_158
Tester:	payalm
Kernel:	3.10.65-b2163-b410+
Audio firmware:	ACF_FWK v1.0 : audio_firmware-ms12_user-37.6.3-0
Video firmware:	Unknown
Linux version:	Linux version 3.10.65-b2163-b410+ (payalm@dlhcxd0052) (gcc version 4.8.3 20141110 (STMicroelectronics/Linux Base 4.8.3-137) (GCC)) #2 SMP PREEMPT Tue Mar 24 12:47:30 IST 2015
Test started at:	2015-04-27 09:56:56
Test ended at:	2015-04-27 10:16:27
Root file:	Using profile: ../../ISTB_tests/sdk2.xml Using profile: ../../ISTB_tests/SystemTest/MediaPlayer/APPS_ISTB_SysTest_MP_root.xml

Detailed report:

Open all test cases
Close all items

Test	Result
TC_SYS_FUNC_PLAYBACK_FILE_LAN_PIP	PASSED
TC_SYS_FUNC_PLAYBACK_FILE_LOCAL_USEROPS_TRICKMODE_MP4_H264_AAC	PASSED
TC_SYS_FUNC_PLAYBACK_FILE_LOCAL_USEROPS_TRICKMODE_AVI_H264_MP3	PASSED
TC_SYS_FUNC_PLAYBACK_FILE_LOCAL_USEROPS_TRICKMODE_FLV_H264_AAC	PASSED
TC_SYS_FUNC_PLAYBACK_FILE_LOCAL_USEROPS_TRICKMODE_MPG_MPEG2_MPEG1	PASSED
TC_SYS_FUNC_PLAYBACK_FILE_LOCAL_WMV_VC1_WMA	PASSED
TC_SYS_FUNC_PLAYBACK_FILE_LOCAL_USEROPS_SEEK_TS_MPEG2_MPEG1	PASSED

Figure 8.8: Results for MediaPlayer testing

Conclusion

Interactive Set-top Box (ISTB) gives flexibility of both STB and Gateway. Both frontend(Gateway) and backend(STB) are independent and self-sufficient. ISTB software is integrated by combining various software components using unified methodology. Sanity testing is one of the testing type which is majorly executed to fix minor bugs/issues to make system more mature.

Validation of ISTB software system includes checking various use cases like functional, parallel, stress and robustness. Validation process of ISTB software system is performed at the end of development process where vast number of cases are performed like LinearTV, ZAP, MediaPlayer, Data browsing etc. This long validation process is automated using TestFramework developed by STMicroelectronics which uses xml and python to accomplish the purpose. Use cases are written in xml files which call python files which performs actual use case on target board which make the process more efficient.

References

[1] ST Internal Document

[2] *Inside Set-top Box, EFY Linux*,by Amit Goel

[3] White Paper on *MPEG-2 Transport vs. Program Stream* by vbricks

[4] http://www.afterdawn.com/glossary/term.cfm/container

[5] *Software Engineering*, by K.K. Aggrawal, Yogesh Singh

[6] http://www.softwaretestingclass.com/smoke-testing/

[7] *Pro Git Book*, by Scott Chacon and Ben Straub

[8] *An Implementation of Interactive Set-Top-Box and Its Applications*, Bing Hu, GuoBin Wu, Liang Pan, Hong Ni, Ming Zhu

[9] *The Unified Software Development Process*, Ivar Jacobson, Grady Booch, James Rumbaugh

[10] *Future capability of cable networks for superfast broadband*, Rod Parker, Alex Slinger, Malcolm Taylor, Matt Yardley