B. TECH. PROJECT REPORT

On

Condition Based Monitoring System

BY

Ayush Soni

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

Nov 2018

Condition Based Monitoring System

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees of
BACHELOR OF TECHNOLOGY
in
Electrical ENGINEERING

Submitted by:
Ayush Soni

Guided by:
Dr. Santosh Kumar Vishwakarma
Associate Professor
Discipline of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE
Nov 2018

CANDIDATE'S DECLARATION

We hereby declare that the project entitled Condition Based Monitoring System submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Ayush Soni completed under the supervision of Dr. Santosh Kumar Vishwakarma, Associate Professor, Discipline of Electrical Engineering, IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Signature and name of the student(s) with date

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

Signature of BTP Guide(s) with dates and their designation

Table of Contents

Preface

This report on "**Condition Based Monitoring System**" is prepared under the guidance of **Dr. Santosh Kumar Vishwakarma, Associate Professor, Discipline of Electrical Engineering, IIT Indore**.

In this internship report I will describe my experiences during my internship period. The internship report contains an overview of the internship company and the activities, tasks and projects that I have worked on during my internship. Writing this report, I also will describe and reflect my learning objects and personal goals that I have set during my internship period. I have tried to discover the relationship between theoretical and practical type of knowledge and to bridge the gap between theoretical assumptions and practical necessities. With these objectives, I have made all possible efforts and the necessary investigations to submit this paper in an enlightened form in a very short time. I have tried my level best to eliminate errors from the paper. As I had to complete my internship within a short period of time so the study admits its limitations.

The report first shall give an overview of the tasks completed during the period of internship with technical details. Then the results obtained shall be discussed and analyzed. The report shall also elaborate on the future works which can be persuaded as an advancement of the current work. I have tried my best to keep report simple yet technically correct. I hope I succeed in my attempt.

I have gained new knowledge and skills. I achieved several of my learning goals. I got insight into professional practice. I learned the different facets of working within a company.

Acknowledgement

This report has been prepared for the internship that has been done at Analog Device India Private Ltd. in order to study the practical aspect of the course and implementation of the theory in the real field with the purpose of fulfilling the requirements of the course of Electrical Engineering.

The aim of this internship is to be familiar to the practical aspect and uses of theoretical knowledge and clarifying the career goals, so I have successfully completed the internship and compiled this report as the summary and the conclusion that have drawn from the internship experience.

I would like to express my sincere gratitude to my internship coordinator and mentor Dr. Santosh Kumar Vishwakarma who have given their valuable time and given me a chance to learn something. And I would like to thank him for supporting me during the completion and documentation of this internship

I would like to thank Mr. Pravin Kumar Angolkar for providing this opportunity to work in Analog Devices India Pvt. Ltd. who helped me to clutch the rear opportunities to learn the real-world situation. I am also grateful to all member of Analog Garage Team for providing several documents, papers, data, figures and services as well as sharing their experience with me and teaching me different techniques to build my knowledge. Thus, the time in Analog Devices very audacious and supportive to my career through which I have gained valuable work experience that will help definitely makes a favorable impression on me as a prospective future employer.

Ayush Soni
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Abstract

In industrial manufacturing rigorous testing is used to ensure that the delivered products meet their specifications. Mechanical maladjustment or faults often show their presence through abnormal acoustic sound signals. This is the same case in machine assembly – the application domain addressed in this paper. Manual diagnosis based on sound requires extensive experience, and usually such experience is acquired at the cost of reduced production efficiency or degraded product quality due to mistakes in judgments. The acquired experience is also difficult to preserve and transfer and it often gets lost if the corresponding personnel leave the task of testing. We propose herein a Case-Based Reasoning approach to collect, preserve and reuse the available experience for machine diagnosis. This solution enables fast experience transfer and more reliable and informed testing. Sounds from normal and faulty machines are recorded and stored in a case library together with their diagnosis results. Given an unclassified sound signal, the relevant cases are retrieved from the case library as a reference for deciding the fault class of the new case. Adding new classified sound profiles to the case library improves the system's performance. So far, the developed system has been applied to the testing environment for industrial machines. The preliminary results demonstrate that our system is valuable in this application scenario in that it can preserve and transfer the related experience among technicians and shortens the overall testing time.

List of Figures

Chapter 1
Introduction

Often with mechanical machines, the operating sound can indicate the condition, the correctness of settings, or the need for maintenance. For example, many drivers can tell, even without training, that there is something wrong with a car by the operating sound. Industrial drilling machines produce characteristic sounds if there is a misalignment. Trained listeners can even identify the specific nature of the problem by sound alone. Although such skills are relatively simple for humans, the implementation of an equivalent artificial system is not straightforward.

One of the new research interests is sound based fault detection, where they are looking for methods to detect and identify faults in mechanical systems by utilizing only operating sound. The practical goal of this thesis is to implement a system able to detect abnormalities in a sound signal for detecting faults.

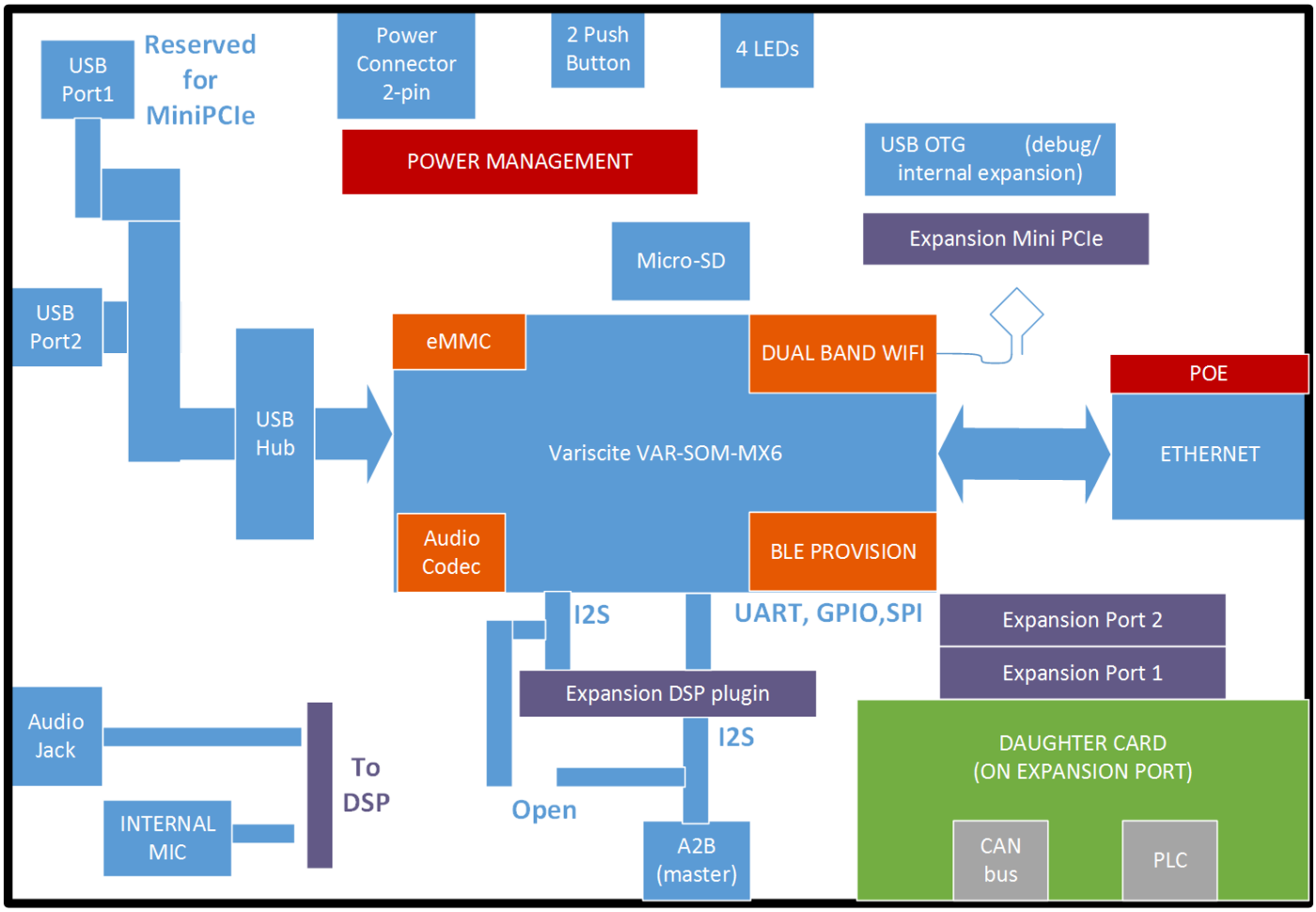


Figure 1.1
Architectural Model of Project

This project is a sensing platform capable of running an algorithm on networked edge nodes and sending the signatures to a central server for analysis and classification. The edge nodes can take multiple streams of data including acoustic and vibration information and send it over the cloud. It is an audio monitoring system implemented on an audio recognition technique. This system provides the untended manufacturing system with the function of overall monitoring capability. It identifies various operational sounds emitted by the machine, and check if they are proper ones which are expected from the control information prior to the

operation. In this project stream of audio data from various sources i.e. audio codec, USB mic, A2B are captured, analyzed and sent to the cloud to detect defect on any machine.

1.2 Problem Statement

The company has devised a machine learning algorithm Condition Based Sound Monitoring System that we had to implement. The goal is to have a system implemented in a concrete device that could warn about impeding faults and later identify specific faults.

1.3 Motivation for work

According to the U.S. Department of Energy, the industrial motor use accounts for 25 percentage of all electricity usage nationwide. Despite the vital roles they play, motors can fail for n number of reasons, leading to a loss in productivity and profitability.

Through the use of AI, sounds can be analyzed to detect machine failure. In other words: Making sound visible even when it can't be heard. With the aid of non-invasive sensors, machine learning algorithms, and predictive maintenance solutions, failing components can be recognized at an early stage before they become a major issue. As sound gets turned into a value, the advancements in hardware and software show that significant progress is being made in the area of audio-based predictive analysis.

Any mechanical machine needs to be maintained intermittently to prevent faults and to ensure that it is operating with normal parameters. Inspite of maintenance, machines can break down for many reasons. When faults begin to appear, a motor's electrical efficiency decreases, hence requiring more energy to compensate for the extra stress that the developing fault is contributing. This creates additional energy consumption and higher electricity bills. This project helps in minimizing energy consumption as well as protecting machines from further damage. Breakdowns and incorrect settings incur costs in several ways - the cost of the physical replacement of the part, the human effort to repair the machine and the lost production time that a machine is not operating. In some cases, the costs can be very high.

The probability of a fault can be lowered with maintenance. In many cases, the maintenance interval is determined statistically and safety margin is used. Avoiding breakages by using high safety margin, leads to unnecessary often done maintenance. If impeding faults could be detected automatically, lower safety margins could be used and downtime of the system can be decreased. Thus, a system that could automatically detect impending faults or wrong settings in a machine before it breaks down, could bring resource savings.

A microphone is a relatively a cheap sensor compared to other kinds of specialized sensors, so it is cost effective and thus it could be applied in many use scenarios. Additionally, hearing is a remote sense thus a microphone can be positioned more freely, unlike many other sensors, like vibration sensors, that require direct access to the machine in question. In addition, there are different hardware platforms that provides high computational capabilities for highly concurrent processing, which are mass producible. Thus such kind of system could be widely applicable and could have many practical industrial and commercial applications.

Chapter 2
Background

Any mechanical system, like a combustion engine, a piping for transporting gaseous or liquid substances or a drill, can incur faults during operation. Faults do not necessarily prevent the system from working at least partially, but if left unchecked can cause the system to breakdown. Faults can occur for many different reasons, like wear down of a part or any defect in a part. Mechanical machines are designed to operate inside tolerance values and in specific conditions. Operating a machine with incorrect settings can lead to a fault causing system breakdown.

System breakdowns are undesired. They can cause danger to living organisms and environment and also incur costs in many ways. For example, a fault in an airplane could potentially cause large number of human casualties. Fixing a machine that has already broken and potentially have caused additional damage due the breakdown requires materials and manpower. In addition, often a system working non-optimally consumes more resources, produces less output and furthers wear down of a machine. For example, a combustion engine operating with incorrect settings consumes more fuel than a correctly set one in order to produce the same output. Additionally, downtime of the system or machine can incur huge costs, especially when the machine is a critical part of a larger system. For example, a pump on an oil rig. Even though it itself is a very expensive machine costing tens of thousands of dollars while it is not working, the whole oil drilling operation comes to a standstill, which incurs losses in the scope of hundreds of millions dollars per day.

There are two kinds of faults-

- **Sudden Faults** that cause immediate system breakdown, caused by breaking of a critical part of the machine, like a crankshaft in a car.
- **Gradual Faults** which allow the system to operate at least partially while causing incremental damage to the system before incurring complete breakdown. Incorrectly set system can be considered as one of the fault categories, which allows the system to operate, but affects the efficiency of the system or the amount of wear that system experiences during operation. Incorrect settings do not necessarily lead to system breakdown but can be costly.
  There are three approaches to address the problem:
  - Preventing Fault
  - Detecting Faults
  - Designing Fault tolerant systems.

Each approach has drawbacks, but these methods can be applied together. For example, utilizing all three approaches in an airplane operation has led to relatively few accidents that happen due to faulty equipment which further results in human casualties. Airplanes are maintained with conservative safety margins: they have redundant systems to allow the plane to operate even if a fault happens and fault detection systems are utilized to warn the operators about impeding faults.

2.1 Fault prevention

Electro-mechanical systems are maintained periodically to prevent faults and to insure correctness of the settings. The required maintenance period is determined statistically so that the machine is expected to function properly between maintenance periods. Safety margins and cost-of-breakdown analysis is utilized to take into account the consequences of a breakdown.

The system maintenance requires resources and operational downtime thus incurring costs. There are many situations where the machine will have to be taken apart in order to inspect its condition. Dismantling a machine for inspection can be time consuming. In addition, in some cases dismantling a machine is impractical, like in case of large industrial heat exchangers. In addition are structural defects that are not apparent to outward inspection. In order to detect structural defects, stress tests can be utilized to verify structural integrity of the part. Additionally, there are non-invasive methods to inspect the structure of a part. The common methods include X-ray and ultrasound imaging techniques.

Additionally, machines can incur faults in spite of being maintained. For example, human error, faulty parts and unexpected conditions. Thus, fault detection schemes could be utilized together with fault prevention to achieve safer and more cost-effective solution.

2.2 Fault Detection

In literature, many various methods are used in non-intrusive fault detection. Some methods are problem specific and act like monitors. For example, dedicated gas detection sensors, which are used in detecting leaks in gas pipes. Fault detection methods are designed either to detect deteriorating operating condition or identifying specific faults during operation time. These methods utilize different signals that an operating machine produces. Some methods rely on specific knowledge about the problem domain.

There are two approaches for detecting faults. One approach utilizes known fault signals and identifies specific faults. The second approach utilizes signals of normal operation in order to find differences that could indicate abnormal operating state.

Methods utilizing known fault signals have a set of pre-gathered samples of signals that represent different fault conditions and the normal condition. The methods utilize a classifier that classifies operating time signals based on the pre-learned set. These methods have advantage over general methods in detecting specific faults, especially if the fault is presented itself consistently in distinct patterns. Additionally, in many cases they are tolerant for external noise, which is due to the approach of finding specific patterns occurring in a signal. Disadvantages are that these methods cannot detect previously unknown faults and they require samples of specific fault conditions. Usually gathering a comprehensive set of fault signals is difficult due irregular nature of fault occurrence. For example, we can think of gathering signals from a high voltage machine where the normal operational time is very long compared to fault occurrences. Moreover, some faults do not produce consistent signals that could be classified properly.

The methods that utilize the second approach, concentrate on finding differences in the signal that is not present in normal conditions. They classify signals based on distance between normal operating sound and the target signal. As advantage, these methods are more general, thus it is easier to adapt them to different use scenarios without having to gather information beforehand. Additionally, they can detect previously unknown faults. Moreover, they are more adept to detect fuzzy occurrences, like squeaking sounds. On the other hand, they have limitations and disadvantages. Firstly, they cannot ignore the operating context. Many machines produce different signals during different states of operation like signals during acceleration are different than operating at constant speed. Secondly, they are more susceptible to external noise signals.

Operating state and condition of a machine can be deduced from many different kind of signals. With the electrical machines, current can be used as a measure to detect physical faults also. In their study, Gong measured current signals to detect faults in Direct-Drive wind turbines. The motor current signature was also used in the study conducted by Khairnar, where they use motor current signature as measure to detect stator faults in three phase induction motors. They use artificial neural network (ANN) to classify the condition of the motor to operate normally, operating under a load, and experiencing fault. Utilizing the operating current as a measure, has drawbacks. It requires direct access to the internal systems of the machine and it is only limited to electrical machines that have windings, like electrical motors and power generators. Vibration is also commonly used in fault detection, as many machines produce vibration signals

during operation. Vibration can be detected by sensors placed on the machine. Vibration signals are significantly similar to sound signals. In the study conducted by Ngolah, they used vibration signal for detecting faults in rotating motors, like a bench grinder. They collected the signals from five piezoelectric transducers that were placed on the machine. An ANN was used to recognize different faults. Alameh utilized vibration to detect faults in permanent magnet synchronous motors. They gathered multiple features both time and frequency domain, like Root Mean-square (RMS) of the signal and mean frequency.

2.3 Sound Based Fault Detection

In addition to sight, the sense of sound is other remote sense that carries information about remote events. Unlike sight, hearing is not limited by field of vision. Mechanical systems commonly produce sound while operating. The operating sounds carry information about the state of the system. Thus, sound signals can be potentially utilized in fault detection. Recently there has been growing interest to use sound-based fault detection methods. Systems that are based on the sound have advantages over other methods. Firstly, they do not require direct contact to the machine, like current or vibration-based methods. Secondly, they are can detect faults that do not present as vibration, like leaking pipes. In addition, microphones are relatively cheap sensors, thus they can be applied cost effectively. Moreover, fiber optic microphones can be utilized in environments with high electromagnetic interference. Sound based methods have a major drawback, they are susceptible to background noise. There are methods that can alleviate the effect of background noise, like noise dampening enclosures.

Chapter 3
Hardware Specification

This section contains specification of various hardware component that had been used in this project.

3.1 Board specification

The VAR-MX6CustomBoard is a single board-computer, utilizing all the VAR-SOM-MX6 System-on-Module features. For development and production, the VAR-MX6CustomBoard serves both as a complete development kit and as an end-product, assembled according to your specifications at an optimized price point.

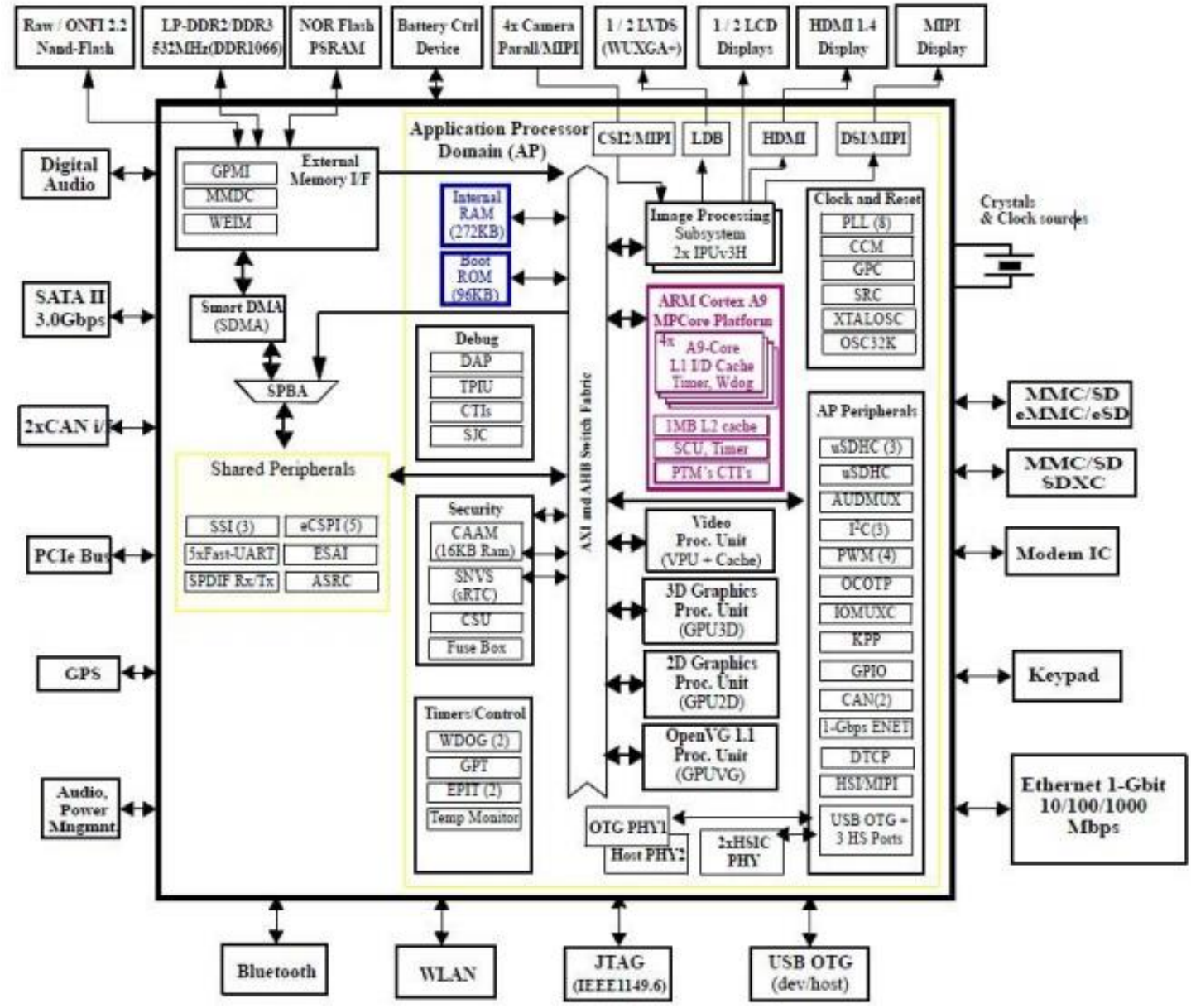


Figure 3.1

System Block Diagram of board

The i.MX6 Dual and i.MX6 Quad processors represent Freescale Semiconductor's latest achievement in integrated multimedia applications processors, optimized for lowest power consumption. The processors feature Freescale's advanced implementation of the quad ARM™ Cortex-A9 core, which operates at speeds of up to 1.2 GHz. They include 2D and 3D graphics processors, 3D 1080p video processing and integrated power management. Each processor provides a 64-bit DDR3/LV-DDR3-1066 memory interface and a number of other interfaces such as WLAN, Bluetooth™, GPS, hard drive, displays, and camera sensors.

3.1.1. Feature Summary

- Freescale i.MX6 series SoC (Single/Dual /Quad ARM® Cortex™-A9 Core, 1.2 GHz)
- Up to 16Gb DDR3 RAM
- 8Gb NAND Flash for storage memory/boot
- 2 x LVDS display interface
- HDMI V1.4 interface
- 1 x MIPI DSI
- Touch panel interface
- Parallel & serial camera interface
- On-board 10/100/1000 Mbps Ethernet PHY
- WLAN (802.11 b/g/n) / BT
- 1 x USB 2.0 host, 1 x OTG
- 2 x SD/MMC
- Serial interfaces (SPI , I2C, UART, I2S,)
- CAN Bus
- Stereo line-In / headphones out
- Digital microphone

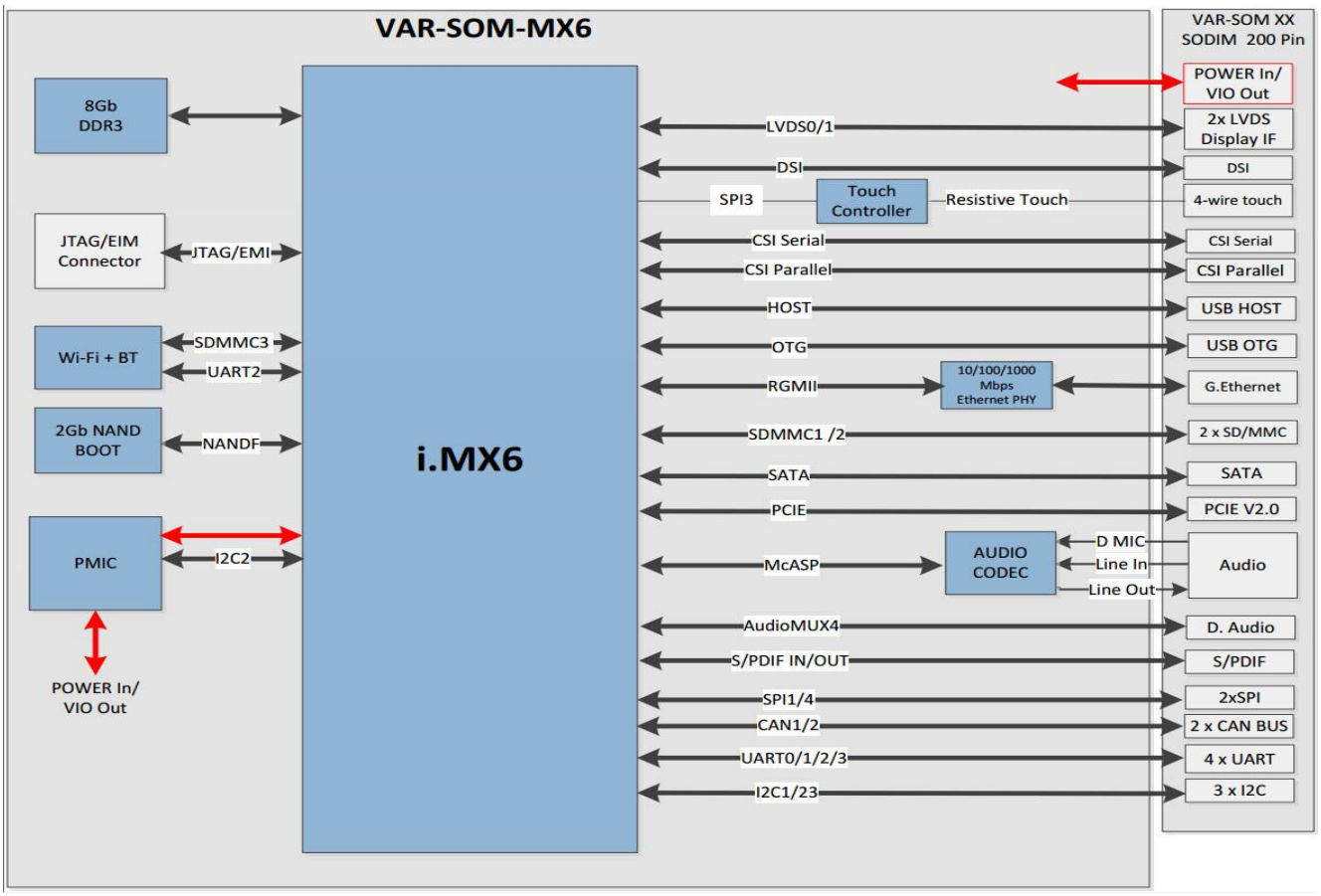


Figure 3.2
Block Diagram of i.MX6

3.1.2. CPU Platform

The i.MX6 Dual / Quad Application Processor (AP) is based on the ARM Cortex-A9 MPCore™ Platform, which has the following features:

- ARM Cortex A9 MPCore™ Dual or Quad core CPU configurations (with TrustZone)
- Symmetric CPU configuration where each CPU includes:
  - 32 Kbyte L1 Instruction Cache
  - 32 Kbyte L1 Data Cache
  - Private Timer and Watchdog
  - Cortex-A9 NEON MPE (Media Processing Engine) Co-processor.
- The ARM Cortex A9 MPCore™ complex includes:
  - General Interrupt Controller (GIC) with 128 interrupt support
  - Global Timer
  - Snoop Control Unit (SCU)
  - 1 Megabyte unified L2 cache shared by all CPU cores (Dual or Quad)
  - Two Master AXI (64-bit) bus interfaces output of L2 cache
- NEON MPE coprocessor
  - SIMD Media Processing Architecture
  - NEON register file with 32x64-bit general-purpose registers
  - NEON Integer execute pipeline (ALU, Shift, MAC)
  - NEON dual, single-precision floating point execute pipeline (FADD, FMUL)
  - NEON load/store and permute pipeline External
  - Supports single and double-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations as described in the ARM VFPv3 architecture.
- Provides conversions between 16-bit, 32-bit and 64-bit floating-point formats and ARM integer word formats.

3.1.3. Memory Interfaces

The memory system consists of the following components:

- Level 1 Cache—32KB Instruction, 32 KB Data cache per core
- Level 2 Cache—Unified instruction and data (1 MByte)
- On-Chip Memory:
  - Boot ROM, including HAB (96 KB)
  - Internal multimedia / shared, fast access RAM (OCRAM, 256 KB)
  - Secure/non-secure RAM (16 KB)
- External memory interfaces:
  - 16-bit, 32-bit, and 64-bit DDR3-1066, LV-DDR3-1066, and 1/2 LPDDR2-1066 channels, supporting DDR interleaving mode, for 2x32 LPDDR2-1066
  - 8-bit NAND-Flash, including support for Raw MLC/SLC, 2 KB, 4 KB, and 8 KB page size,
  - BA-NAND, PBA-NAND, LBA-NAND, OneNAND™ and others. BCH ECC up to 32 bit.
- 16-bit NOR Flash. All WEIMv2 pins are muxed on other interfaces.
- 16-bit PSRAM, Cellular RAM

3.1.4. Audio Back End

The AUDMUX provides flexible, programmable routing of the serial interfaces (SSI1 or SSI2) to and from off-chip devices. The AUDMUX routes audio data (and even splices together multiple time-multiplexed audio streams) but does not decode or process audio data itself. The AUDMUX is controlled by the ARM but can route data even when the ARM is in a low power mode.

The ESAI (Enhanced Serial Audio Interface) provides a full-duplex serial port for serial communication with a variety of serial devices, including industry-standard codecs, SPDIF transceivers, and

other processors. The ESAI consists of independent transmitter and receiver sections, each section with its own clock generator. The ESAI is connected to the IOMUX and to the ESAI_BIFIFO module.

The ESAI_BIFIFO (ESAI Bus Interface and FIFO) is the interface between the ESAI module and the shared peripheral bus. It contains the FIFOs used to buffer data to and from the ESAI, as well as providing the data word alignment and padding necessary to match the 24-bit data bus of the ESAI to the 32-bit data bus of the shared peripheral bus.

The SPDIF (Sony/Philips Digital Interface) audio module is a stereo transceiver that allows the processor to receive and transmit digital audio over it. The SPDIF receiver section includes a frequency measurement block that allows the precise measurement of incoming sampling frequency. A recovered clock is provided by the SPDIF receiver section and may be used to drive both internal and external components in the system. The SPDIF is connected to the shared peripheral bus.

The ASRC (Asynchronous Sample Rate Converter) converts the sampling rate of a signal associated to an input clock into a signal associated to a different output clock. The ASRC supports concurrent sample rate conversions of up to 10 channels of over 120 dB THD+N. The sample rate conversion of each channel is associated to a pair of incoming and outgoing sampling rates. The ASRC supports up to three sampling rate pairs. The ASRC is connected to the shared peripheral bus.

3.1.5. 10/100/1000 Ethernet Controller

The MAC-NET core, in conjunction with a 10/100/1000 MAC, implements layer 3 network acceleration functions. These functions are designed to accelerate the processing of various common networking protocols, such as IP, TCP, UDP and ICMP, providing wire speed services to client applications. The MAC operation is fully programmable and can be used in NIC (Network Interface Card), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819. The core also implements a hardware acceleration block to optimize the performance of network controllers providing IP and TCP, UDP, ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead. The core implements programmable embedded FIFOs that can provide buffering on the receive path for loss-less flow control. Advanced power management features are available with magic packet detection and programmable power-down modes.

3.1.5. Memory

3.1.5.1. RAM

The VAR-SOM-MX6 is available with up to 16Gb of DDR3 memory.

3.1.5.2. Non-volatile Storage Memory

The VAR-SOM-MX6 is available with up to 8Gb of SLC NAND FLASH memory. The NAND flash is used for Flash Disk purposes, O.S. run-time-image and the Boot-loader (Boot from NAND). First block (block address 00h) of the memory device is guaranteed to be valid without ECC (up to 1,000 PROGRAM/ERASE cycles)

3.1.6. 10/100/1000 Ethernet PHY

The VAR-SOM-MX6 features the Micrel KSZ9031 gigabit Ethernet PHY. The KSZ9031RN is a completely integrated triple speed (10Base-T/100Base-TX/1000Base-T) Ethernet Physical Layer Transceiver for transmission and reception of data over standard CAT-5 unshielded twisted pair (UTP) cable. The KSZ9031RN provides the Reduced Gigabit Media Independent Interface (RGMII) for direct connection to RGMII MACs in Gigabit Ethernet processors and switches for data transfer at 10/100/1000 Mbps speed.

3.1.7. TLV320AIC3106 Audio

The Texas Instrument's TLV320AIC3106 is a low-power, highly integrated stereo audio codec with stereo headphone amplifier, as well as multiple inputs and outputs programmable in single-ended or fully differential configurations. Extensive register-based power control is included, enabling stereo 48-kHz DAC playback as low as 15 mW. The VAR-SOM-MX6 exposes the following interface of the TLV320AIC3106:

- Headphone
- Line-in
- Digital microphone

3.1.8. Wi-Fi + BT

Wi-Fi & Bluetooth connectivity is supported by an on-board module TiWi – BLE. The TiWi-BLE module is a high performance 2.4 GHz IEEE 802.11 b/g/n Bluetooth 4.0 radio in a cost effective, pre-certified footprint. The module realizes the necessary PHY/MAC layers to support WLAN applications in conjunction with a host processor over a SDIO interface. The module also provides a Bluetooth platform through the HCI transport layer. Both WLAN and Bluetooth share the same antenna port.

3.2 Audio Codec Specification

The TLV320AIC3106 is a low-power stereo audio codec with stereo headphone amplifier, as well as multiple inputs and outputs programmable in single-ended or fully differential configurations. Extensive register-based power control is included, enabling stereo 48-kHz DAC playback as low as 15 mW from a 3.3-V analog supply, making it ideal for portable battery-powered audio and telephony applications.

The record path of the TLV320AIC3106 contains integrated microphone bias, digitally controlled stereo microphone preamplifier, and automatic gain control (AGC), with mix/mux capability among the multiple analog inputs. Programmable filters are available during record which can remove audible noise that can occur during optical zooming in digital cameras.

The playback path includes mix/mux capability from the stereo DAC and selected inputs, through programmable volume controls, to the various outputs.

The TLV320AIC3106 contains four high-power output drivers as well as three fully differential output drivers. The high-power output drivers are capable of driving a variety of load configurations, including up to four channels of single-ended 16-Ω headphones using ac-coupling capacitors, or stereo 16-Ω headphones in a capacitorless output configuration.

The stereo audio DAC supports sampling rates from 8 kHz to 96kHz and includes programmable digital filtering in the DAC path for 3D, bass, treble, midrange effects, speaker equalization, and de-emphasis for 32kHz, 44.1kHz, and 48kHz rates. The stereo audio ADC supports sampling rates from 8 kHz to 96kHz and is preceded by programmable gain amplifiers or AGC that can provide up to 59.5-dB analog gain for low-level microphone inputs. The TLV320AIC3106 provides an extremely high range of programmability for both attack (8ms - 1,408ms) and for decay (0.05 s–22.4 s). This extended AGC range allows the AGC to be tuned for many types of applications.

For battery saving applications where neither analog nor digital signal processing are required, the device can be put in a special analog signal passthrough mode. This mode significantly reduces power consumption, as most of the device is powered down during this pass-through operation.

The serial control bus supports SPI or I2C protocols, while the serial audio data bus is programmable for I2S, left/right-justified, DSP, or TDM modes. A highly programmable PLL is included for flexible clock generation and support for all standard audio rates from a wide range of available MCLKs, varying from 512 kHz to 50 MHz, with special attention paid to the most popular cases of 12-MHz, 13-MHz, 16-MHz, 19.2-MHz, and 19.68-MHz system clocks.

The TLV320AIC3106 consists of the following blocks:

- Stereo audio multi-bit delta-sigma DAC (8 kHz–96 kHz)
- Stereo audio multi-bit delta-sigma ADC (8 kHz–96 kHz)
- Programmable digital audio effects processing (3-D, bass, treble, mid-range, EQ, notch filter, de-emphasis)
- Six audio inputs
- Four high-power audio output drivers (headphone drive capability)
- Three fully differential line output drivers

- Fully programmable PLL

Headphone/headset jack detection with interrupt Communication to the TLV320AIC3106 for control is pin-selectable (using the SELECT pin) as either SPI or I2C. The SPI interface requires that the Slave Select signal (MFP0) be driven low to communicate with the TLV320AIC3106. Data is then shifted into or out of the TLV320AIC3106 under control of the host microprocessor, which also provides the serial data clock. The I2C interface supports both standard and fast communication modes, and also enables cascading of up to four multiple codecs on the same I2C bus through the use of two pins for addressing (MFP0, MFP1).

Digital Audio Data Serial Interface

Audio data is transferred between the host processor and the TLV320AIC3106 via the digital audio data serial interface, or audio bus. The audio bus on this device is very flexible, including left or right justified data options, support for I2S or PCM protocols, programmable data length options, a TDM mode for multichannel operation, very flexible master/slave configurability for each bus clock line, and the ability to communicate with multiple devices within a system directly. The data serial interface uses two sets of pins for communication between external devices, with the particular pin used controlled through register programming. This configuration is shown in Figure below.

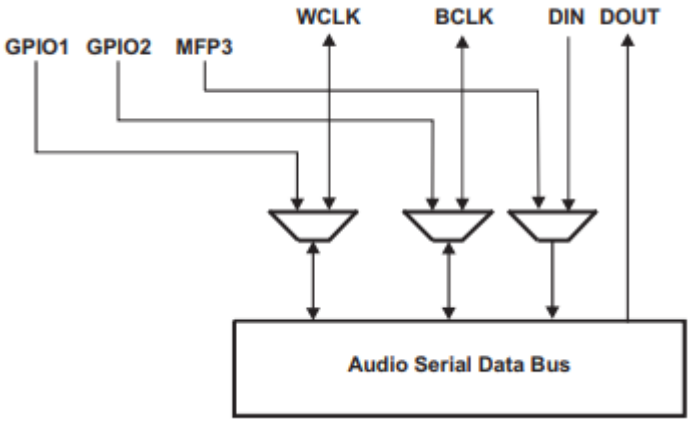


Figure 3.3
Alternate Audio Bus Multiplexing Function

In cases where MFP3 is needed for a secondary device digital input, the TLV320AIC3106 must be used in I2C mode (when in SPI mode, MFP3 is used as the SPI bus MOSI pin and thus cannot be used here as an alternate digital input source). This mux capability allows the TLV320AIC3106 to communicate with two separate devices with independent I2S/PCM buses. An example of such an application is a cell phone containing a Bluetooth transceiver with PCM/I2S interface. The applications processor can be connected to the WCLK, BCLK, DIN, DOUT pins on the TLV320AIC3106, while a Bluetooth device with PCM interface can be connected to the GPIO1, GPIO2, MFP3, and DOUT pins on the TLV320AIC3106. By programming the registers via I2C control, the applications processor can determine which device is communicating with the TLV320AIC3106. This is attractive in cases where the TLV320AIC3106 can be configured to communicate data with the Bluetooth device, then the applications processor can be put into a low power sleep mode, while voice/audio transmission still occurs between the Bluetooth device and the TLV320AIC3106.

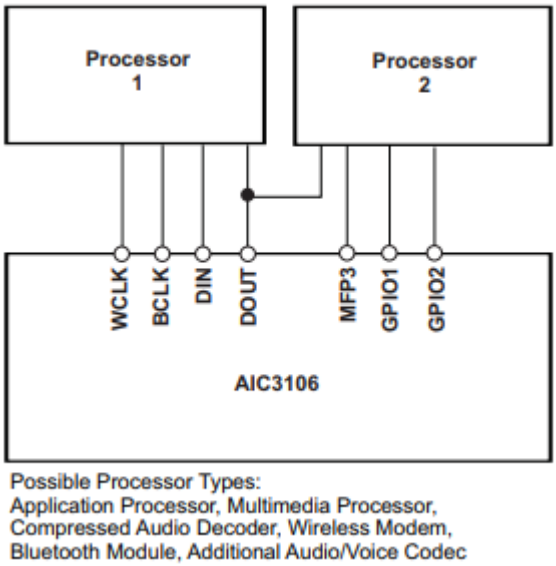


Figure 3.4
Codec's Connection to Multiple Audio Devices

The audio bus of the TLV320AIC3106 can be configured for left or right justified, I2S, DSP, or TDM modes of operation, where communication with standard telephony PCM interfaces is supported within the TDM mode. These modes are all MSB-first, with data width programmable as 16, 20, 24, or 32 bits. In addition, the word clock (WCLK or GPIO1) and bit clock (BCLK or GPIO2) can be independently configured in either Master or Slave mode, for flexible connectivity to a wide variety of processors

The word clock (WCLK or GPIO1) is used to define the beginning of a frame, and may be programmed as either a pulse or a square-wave signal. The frequency of this clock corresponds to the maximum of the selected ADC and DAC sampling frequencies.

The bit clock (BCLK or GPIO2) is used to clock in and out the digital audio data across the serial bus. When in Master mode, this signal can be programmed in two further modes: continuous transfer mode, and 256-clock mode. In continuous transfer mode, only the minimal number of bit clocks needed to transfer the audio data are generated, so in general the number of bit clocks per frame will be two times the data width. For example, if data width is chosen as 16 bits, then 32-bit clocks will be generated per frame. If the bit clock signal in master mode will be used by a PLL in another device, it is recommended that the 16-bit or 32-bit data width selections be used. These cases result in a low jitter bit clock signal being generated, having frequencies of 32 × fs or 64 × fs. In the cases of 20-bit and 24-bit data width in master mode, the bit clocks generated in each frame will not all be of equal period, due to the device not having a clean 40 × fs or 48 × fs clock signal readily available. The average frequency of the bit clock signal is still accurate in these cases (being 40 × fs or 48 × fs), but the resulting clock signal has higher jitter than in the 16-bit and 32-bit cases.

In 256-clock mode, a constant 256-bit clocks per frame are generated, independent of the data width chosen. The TLV320AIC3106 further includes programmability to 3-state the DOUT line during all bit clocks when valid data is not being sent. By combining this capability with the ability to program at what bit clock in a frame the audio data will begin, time-division multiplexing (TDM) can be accomplished, resulting in multiple codecs able to use a single audio serial data bus.

Chapter 4
Protocols

A protocol is a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two or more networks or devices, they have to follow these rules to successfully transmit data.

Protocols specify the standards for communication and provide detailed information on processes involved in data transmission. Such processes include:

- Type of task
- Process nature
- Data flow rate
- Data type
- Device management

4.1 Electronic Communication Protocols

Electronic Communication protocols are formal descriptions of digital message formats and rules. They are required to exchange messages in or between computing systems and are required in telecommunications.

Communications protocols cover authentication, error detection and correction, and signaling. They can also describe the syntax, semantics, and synchronization of analog and digital communications. Communications protocols are implemented in hardware and software. There are thousands of communications protocols that are used everywhere in analog and digital communications.

Communications devices have to agree on many physical aspects of the data to be exchanged before successful transmission can take place. Rules defining transmissions are called protocols.

Types of Electronic Communication Protocols:

There are two types of communication product which are classified below:

1. Inter System Protocol
2. Intra System Protocol

1. **Inter System Protocol:** The inter system protocol used to communicate the two different devices. Like communication between to microcontroller kit. The communication is done through a inter bus system.

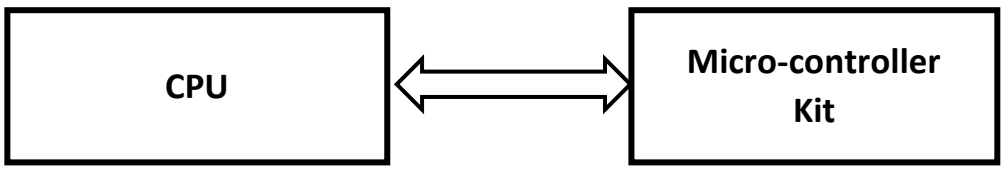


Figure 4.1
Inter System Protocol

Different Categories of Inter System Protocols:

- UART Protocol
- USART Protocol
- USB Protocol

2. **Intra System Protocol:** The Intra system protocol is used to communicate the two devices within the circuit board. While using this intra system protocols, without going to intra system protocols we will expand the peripherals of the microcontroller. The circuit complexity and power consumption will be increase by using intra system protocols. Using intra system protocols circuit complexity and power consumption, cost is decrease and it is very secure to accessing the data.

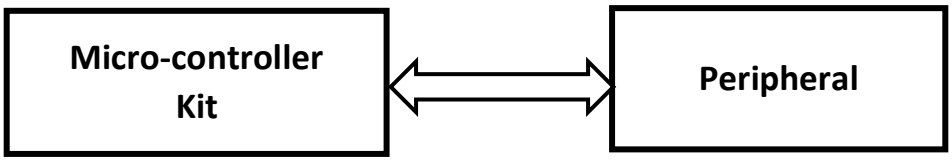


Figure 4.2
Intra System Protocol

Different categories of Inter system protocol

- I2C Protocol
- SPI Protocol
- CAN Protocol

4.1.1 Universal Asynchronous Communication Transfer Protocol

A **universal asynchronous receiver-transmitter** (**UART**) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. The electric signaling levels and methods are handled by a driver circuit external to the UART. A UART is usually an individual (or part of an) integrated circuit (IC) used for serial communication over a computer or peripheral device serial port. In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Data flows from the TX pin of the transmitting UART to the RX pin of the receiving UART:

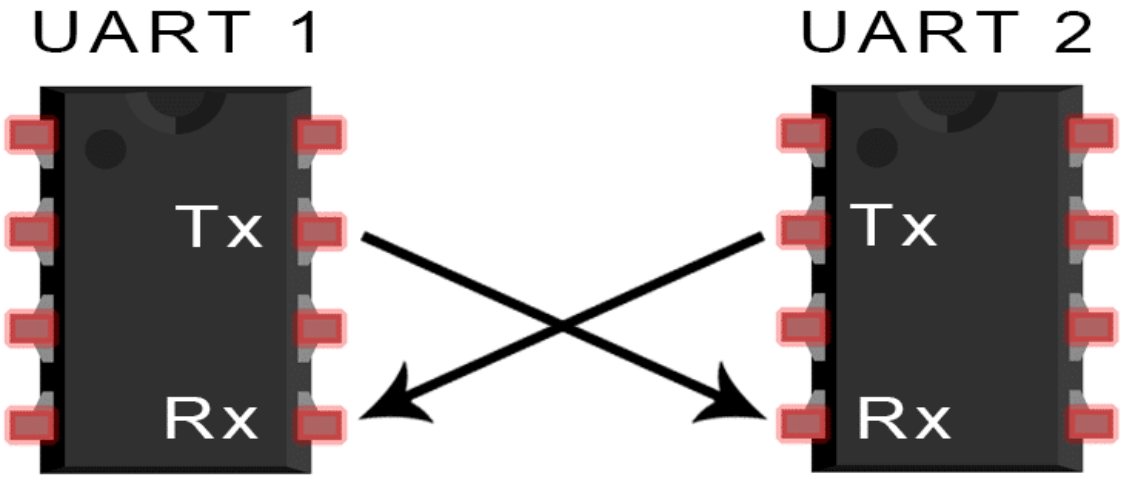


Figure 4.3
Connection between two UART devices

UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART add start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits. When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off.

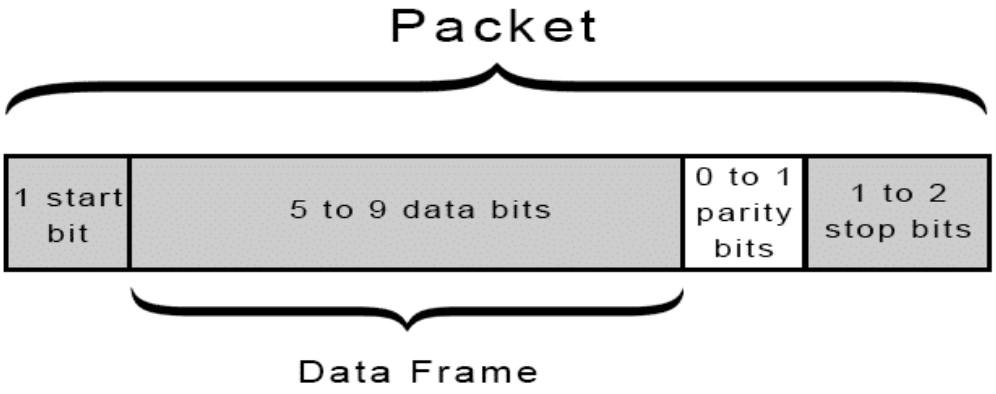


Figure 4.4
Data Packet Format of UART

START BIT
The UART data transmission line is normally held at a high voltage level when it's not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.

DATA FRAME
The data frame contains the actual data being transferred. It can be 5 bits up to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. In most cases, the data is sent with the least significant bit first.

PARITY
Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission. Bits can be changed by electromagnetic radiation, mismatched baud rates, or long-distance data transfers. After the receiving UART reads the data frame, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. If the parity bit is a 0 (even parity), the 1 bit in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bit in the data frame should total to an odd number. When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd; or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed.

STOP BITS
To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two-bit durations.

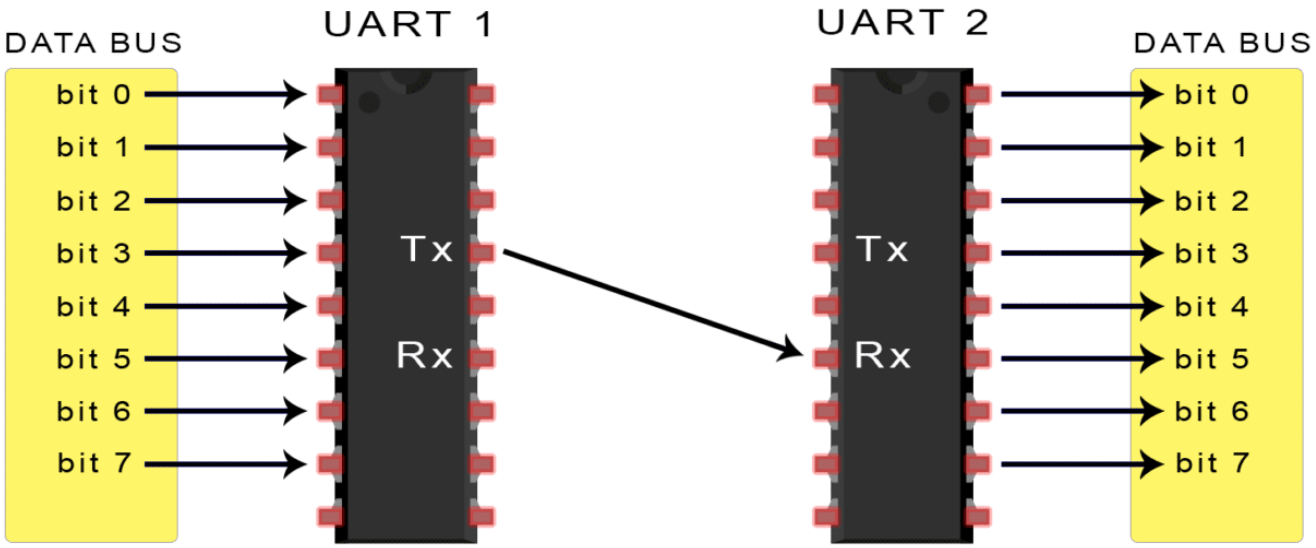


Figure 4.5
UART communication between two devices

Theory of Operation

1. The transmitting UART receives data in parallel from the data bus.
2. The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame.
3. The entire packet is sent serially from the transmitting UART to the receiving UART. The receiving UART samples the data line at the pre-configured baud rate.
4. The receiving UART discards the start bit, parity bit, and stop bit from the data frame.
5. The receiving UART convert the serial data back into parallel and transfer it to the data bus on the receiving end.

4.1.2 Inter-IC Communication Protocol

Inter-Integrated Circuit (I²C) is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus. It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication.I2C only uses two wires to transmit data between devices:

SDA (Serial Data) – The line for the master and slave to send and receive data.

SCL (Serial Clock) – The line that carries the clock signal.

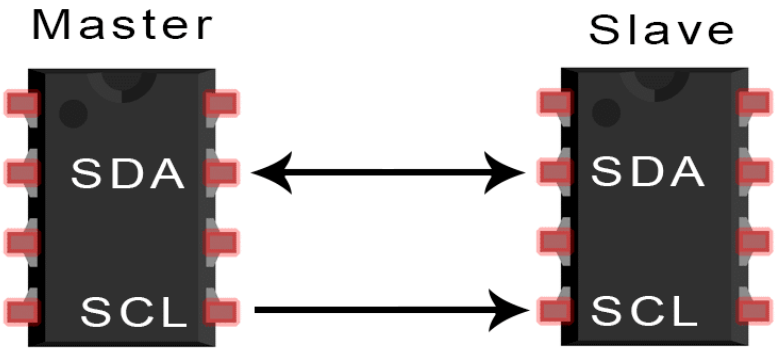


Figure 4.6
Connection between two I2C devices

With I2C, data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame:

Start Condition:

The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.

Stop Condition:

The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.

Address Frame:

A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.

Read/Write Byte:

A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).

Acknowledgement Bit:

Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

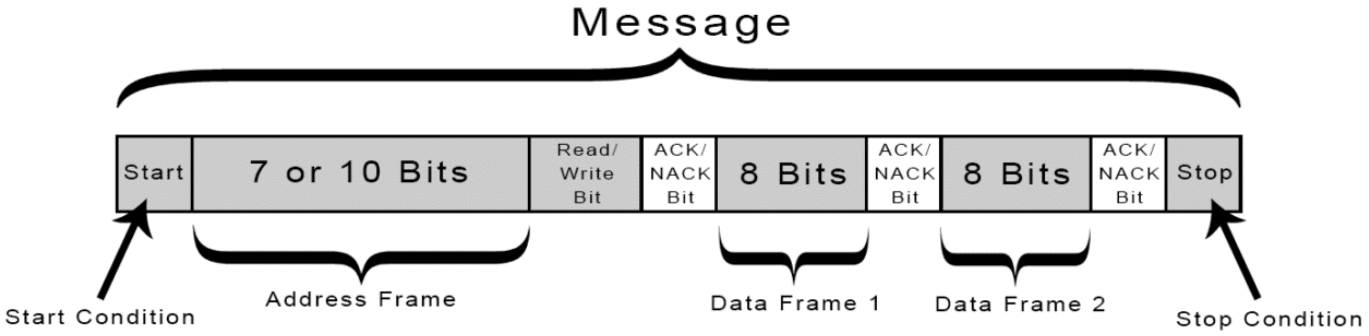


Figure 4.7
Data Packet Format of I2C

Theory of Operation

1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low.
2. The master sends each slave the 7- or 10-bit address of the slave it wants to communicate with, along with the read/write bit.
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the SDA line high.
4. The master sends or receives the data frame.
5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.
6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high.

4.1.3 Inter-IC Sound Protocol

I²S (**Inter-IC Sound**) is an electrical serial bus interface standard used for connecting digital audio devices together. It is used to communicate PCM audio data between integrated circuits in an electronic device. 2S is a simple data interface, without any form of address or device selection. On an I2S bus, there is only one bus master and one transmitter. The master may be a transmitter, a receiver, or a controller for data transfers between other devices acting as transmitter and receiver. In high-quality audio applications involving a codec, the codec is typically the master so that it has precise control over the I2S bus clock. The I2S bus carries two channels, left and right, which are typically used to carry stereo audio data streams. The data alternates between left and right channels, as controlled by a word select signal driven by the bus master.

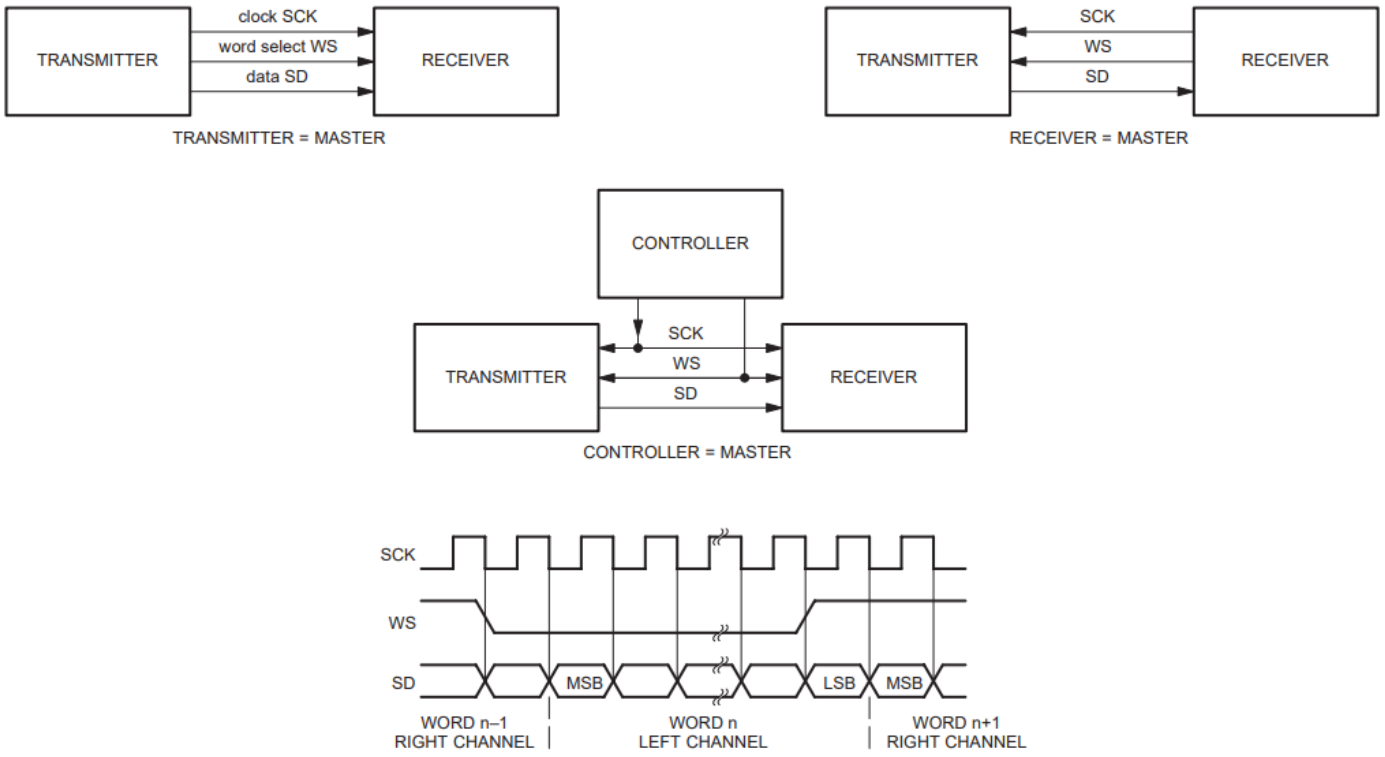


Figure 4.8
Different Modes of I2S

I²S (**Inter-IC Sound**) bus has 3 lines:

- **Continuous Serial Clock (SCK)**
- **Word Select (WS) or Left-Right Select Clock (LRCLK)**
- **Serial Data (SD)**

Master generates SCK and WS in this communication protocol.

Serial Data (SD)

Serial data is transmitted in two's complement with the MSB first. The MSB is transmitted first because the transmitter and receiver may have different word lengths. It isn't necessary for the transmitter to know how many bits the receiver can handle, nor does the receiver need to know how many bits are being transmitted. When the system word length is greater than the transmitter word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally. And so, the MSB has a fixed position, whereas the position of the LSB depends on the word length. The transmitter always sends the MSB of the next word one clock period after the WS changes. Serial data sent by the transmitter may be synchronized with either the trailing

(HIGH-to-LOW) or the leading (LOW-to-HIGH) edge of the clock signal. However, the serial data must be latched into the receiver on the leading edge of the serial clock signal, and so there are some restrictions when transmitting data that is synchronized with the leading edge

Word Select (WS)

The word select line indicates the channel being transmitted:

WS = 0 - channel 1 (Left channel)

WS = 1 - channel 2 (Right channel)

WS may change either on a trailing or leading edge of the serial clock, but it doesn't need to be symmetrical. In the slave, this signal is latched on the leading edge of the clock signal. The WS line changes one clock period before the MSB is transmitted. This allows the slave transmitter to derive synchronous timing of the serial data that will be set up for transmission. Furthermore, it enables the receiver to store the previous word and clear the input for the next word.

4.2 Sockets Protocols

Socket - A socket is a software object that acts as an end point establishing a bidirectional network communication link between a server-side and a client-side program. In UNIX, a socket can also be referred to as an endpoint for Inter-process communication (IPC) within the operating system (OS). One socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

Socket Types

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- **Stream Sockets**
- **Datagram Sockets**
- **Raw Sockets**
- **Sequenced Packet Sockets**

Socket protocols

A protocol is a standard set of rules for transferring data, such as UDP/IP and TCP/IP. An application program can specify a protocol only if more than one protocol is supported for this particular socket type in this domain.
Each socket can have a specific protocol associated with it. This protocol is used within the domain to provide the semantics required by the socket type. Not all socket types are supported by each domain; support depends on the existence and implementation of a suitable protocol within the domain.

Major Protocols in the suite of Internet Network Protocols include:

- Transmission Control Protocol/Internet Protocol
- User Datagram Protocol
- Reliable Datagram Socket Protocol
- Internet Control Message Protocol

4.2.1 User Datagram Protocol

UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection.

Features

- UDP is used when acknowledgement of data does not hold any significance.
- UDP is good protocol for data flowing in one direction.
- UDP is simple and suitable for query-based communications.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.

UDP Header

UDP header is as simple as its function.

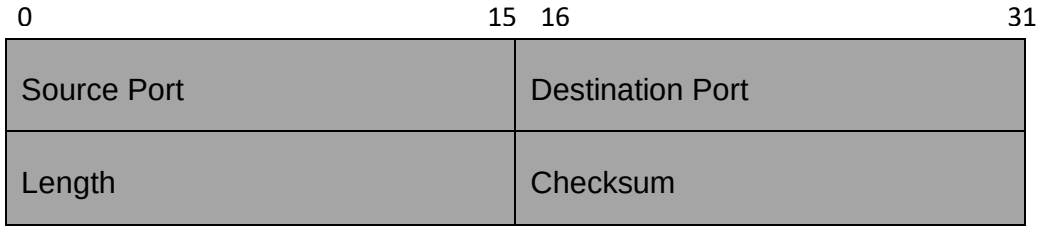


Figure 4.9
UDP Data Packet Format

UDP header contains four main parameters:

- **Source Port** - This 16 bits information is used to identify the source port of the packet.
- **Destination Port** - This 16 bits information, is used identify application level service on destination machine.
- **Length** - Length field specifies the entire length of UDP packet (including header). It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header itself.
- **Checksum** - This field stores the checksum value generated by the sender before sending. IPv4 has this field as optional so when checksum field does not contain any value it is made 0 and all its bits are set to zero.

4.2.2 Transmission Control Protocol

The Internet protocol or Transmission Control Protocol is the conceptual model and set of communication protocols used on the Internet and similar computer networks.

The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed, and received. TCP ensures reliability by sequencing bytes with a forwarding acknowledgement number that indicates to the destination the next byte the source expect to

receive. TCP offers Stream Data Transfer, Reliability, Efficient Flow Control, Full-duplex operation and Multiplexing.

This functionality is organized into four abstraction layers, which classify all related protocols according to the scope of networking involved. From lowest to highest, the layers are the link layer, containing communication methods for data that remains within a single network segment ; the internet layer, providing internetworking between independent networks; the transport layer, handling host-to-host communication; and the application layer, providing process-to-process data exchange for applications.

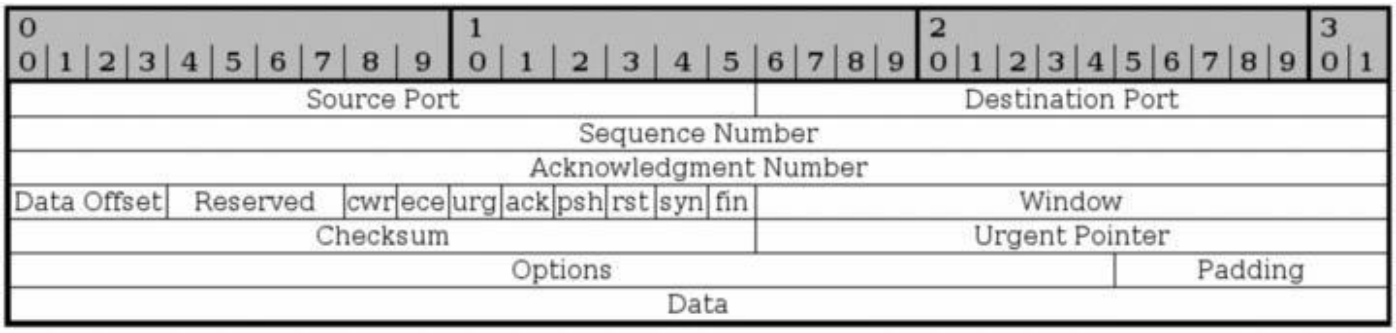


Figure 4.10
TCP Data Packet Format

TCP Services

TCP offers following services to the processes at the application layer:

- Stream Delivery Service
- Sending and Receiving Buffers
- Bytes and Segments
- Full Duplex Service
- Connection Oriented Service
- Reliable Service

STREAM DELIVER SERVICE

TCP protocol is stream oriented because it allows the sending process to send data as stream of bytes and the receiving process to obtain data as stream of bytes.

SENDING AND RECEIVING BUFFERS

It may not be possible for sending and receiving process to produce and obtain data at same speed, therefore, TCP needs buffers for storage at sending and receiving ends.

BYTES AND SEGMENTS

The Transmission Control Protocol (TCP), at transport layer groups the bytes into a packet. This packet is called segment. Before transmission of these packets, these segments are encapsulated into an IP datagram.

FULL DUPLEX SERVICE

Transmitting the data in duplex mode means flow of data in both the directions at the same time.

CONNECTION ORIENTED SERVICE

TCP offers connection-oriented service in the following manner:

1. TCP of process-1 informs TCP of process-2 and gets its approval.
2. TCP of process-1 and TCP of process-2 and exchange data in both the two directions.
    1. After completing the data exchange, when buffers on both sides are empty, the two TCP's destroy their buffers.

RELIABLE SERVICE

For sake of reliability, TCP uses acknowledgement mechanism.

TCP model contains four layers, which are:

1. Process/Application Layer
2. Host-to-Host/Transport Layer
3. Internet Layer
4. Network Access/Link Layer

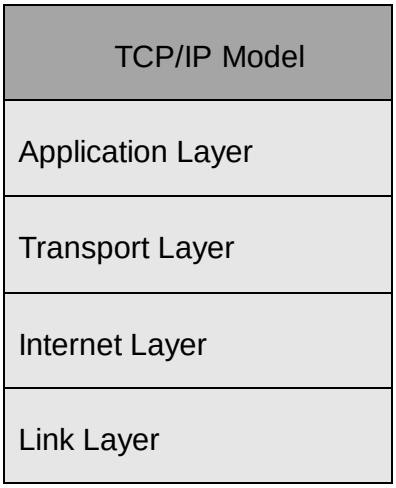


Figure 4.11
Different Layers of TCP Model

Application layer

The application layer is the scope within which applications create user data and communicate this data to other applications on another or the same host. The applications, or processes, make use of the services provided by the underlying, lower layers, especially the Transport Layer which provides reliable or unreliable pipes to other processes. The communications partners are characterized by the application architecture, such as the client-server model and peer-to-peer networking. This is the layer in which all higher-level protocols, such as SMTP, FTP, SSH, HTTP, operate. Processes are addressed via ports which essentially represent services.

Transport layer

The transport layer performs host-to-host communications on either the same or different hosts and on either the local network or remote networks separated by routers. It provides a channel for the communication needs of applications.

Internet layer

The internet layer exchanges datagrams across network boundaries. It provides a uniform networking interface that hides the actual topology (layout) of the underlying network connections. It is therefore also referred to as the layer that establishes internetworking. Indeed, it defines and establishes the Internet. This layer defines the addressing and routing structures used for the TCP/IP protocol suite.

Link layer

The link layer defines the networking methods within the scope of the local network link on which hosts communicate without intervening routers. This layer includes the protocols used to describe the local network topology and the interfaces needed to effect transmission of Internet layer datagrams to next-neighbor hosts.

4.2.3 Internet Control Message Protocol

ICMP (Internet Control Message Protocol) is an error-reporting protocol network device like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets. ICMP creates and sends messages to the source IP address indicating that a gateway to the Internet that a router, service or host cannot be reached for packet delivery. Any IP network device has the capability to send, receive or process ICMP messages. It is used by network administrators to troubleshoot Internet connections in diagnostic utilities including ping and traceroute. One of the main protocols of the Internet Protocol suite, ICMP is used by routers, intermediary devices or hosts to communicate error information or updates to other routers, intermediary devices or hosts. The widely used IPv4 (Internet Protocol version 4) and the newer IPv6 use similar versions of the ICMP protocol (ICMPv4 and ICMPv6, respectively).

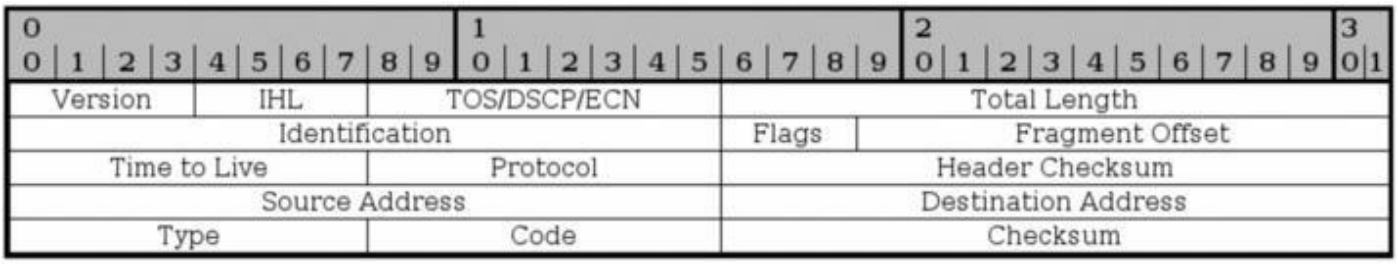


Figure 4.12
ICMP Data Packet Format

4.3 Network Time Protocol

The **Network Time Protocol** (**NTP**) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. NTP is intended to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC).[1]:3 It uses the intersection algorithm, a modified version of Marzullo's algorithm, to select accurate time servers and is designed to mitigate the effects of variable network latency. NTP can usually maintain time to within tens of milliseconds over the public Internet, and can achieve better than one millisecond accuracy in local area networks under ideal conditions. The protocol is usually described in terms of a client-server model, but can as easily be used in peer-to-peer relationships where both peers consider the other to be a potential time source. Implementations send and receive timestamps using the User Datagram Protocol (UDP) on port number 123. They can also use broadcasting or multicasting, where clients passively listen to time updates after an initial round-trip calibrating exchange. NTP uses a systematic, hierarchical level of clock sources for its reference. Each level is called a stratum and has a layer number that usually begins with zero. The stratum level serves as an indicator of the distance from the reference clock in order to avoid cyclic dependency in the hierarchy. However, the stratum does not represent the quality or reliability of time.

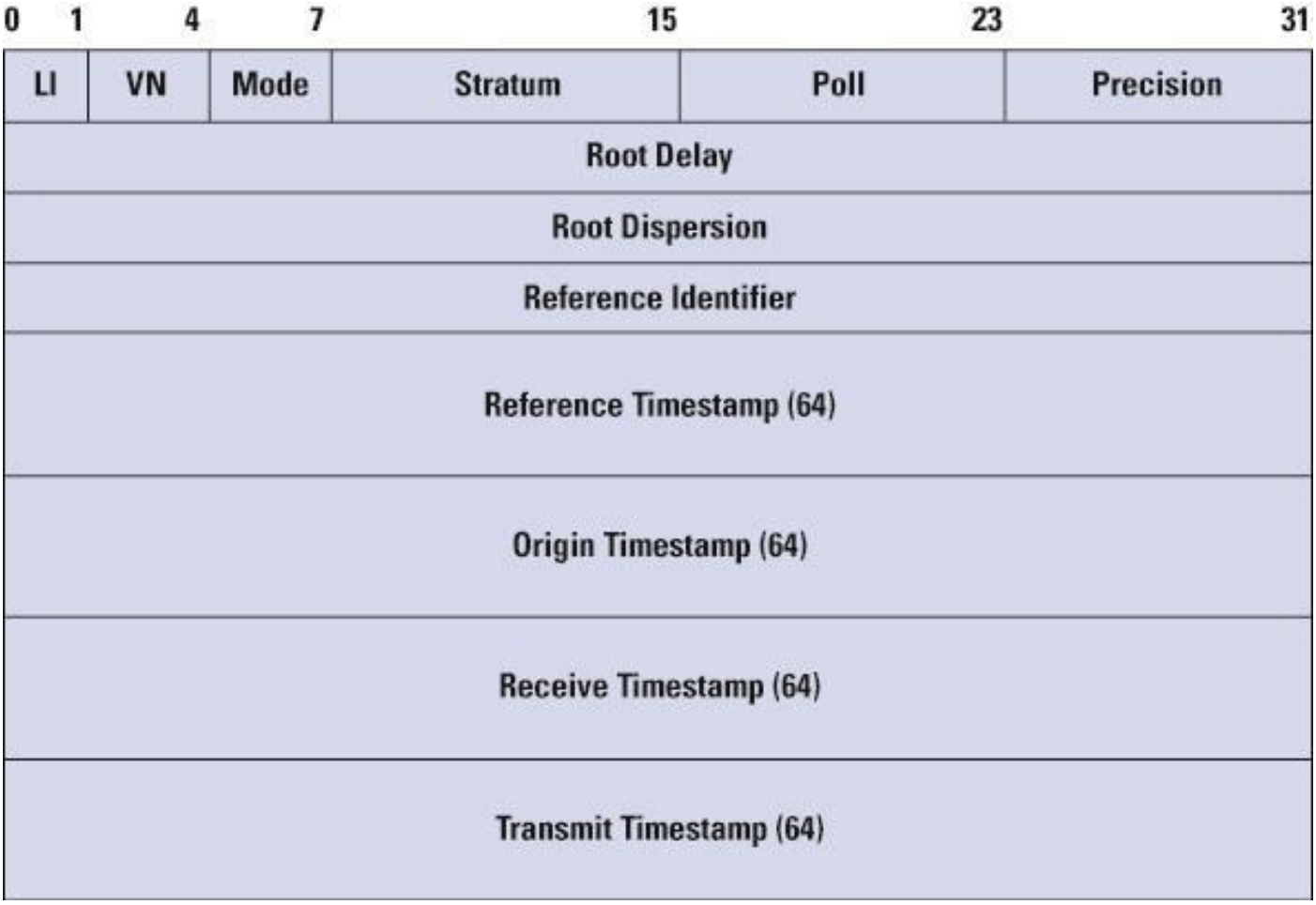


Figure 4.13
NTP Data Packet Structure

4.4 ALSA Framework

ALSA stands for the Advanced Linux Sound Architecture Framework. It consists of a set of kernel drivers, an application programming interface (API) library and utility programs for supporting sound card devices under Linux.

ALSA Kernel Space Features:

- Efficiently and fully supports from consumer sound cards to professional multichannel audio interfaces, bringing features not supported by OSS, such as hardware-based MIDI synthesis, software mixing of multiple channels and full duplex operation.
- Supports SMP (multiprocessor) systems. Thread-Safe device drivers and user space library.
- Consistent and generic control API for managing hardware controls.

The ALSA API can be broken down into the major interfaces it supports:

- Control interface: a general-purpose facility for managing registers of sound cards and querying the available devices.
- PCM interface: the interface for managing digital audio capture and playback. The rest of this article focuses on this interface, as it is the one most commonly used for digital audio applications.
- Raw MIDI interface: supports MIDI (Musical Instrument Digital Interface), a standard for electronic musical instruments. This API provides access to a MIDI bus on a sound card. The raw interface works directly with the MIDI events, and the programmer is responsible for managing the protocol and timing.

- Timer interface: provides access to timing hardware on sound cards used for synchronizing sound events.
- Sequencer interface: a higher-level interface for MIDI programming and sound synthesis than the raw MIDI interface. It handles much of the MIDI protocol and timing.
- Mixer interface: controls the devices on sound cards that route signals and control volume levels. It is built on top of the control interface.

Chapter 5
Contribution

In this project, which is a Condition based monitoring platform, I made different modules for capturing acoustic audio data, provisioning of i.MX6 board to configure and capture signal path of audio codec, a self-test module and a bug logger module.

5.1 Audio Capture Module

In this module, my work was to capture the audio data from Line-In and Digital Mic which are on the Audio Codec (TLV320aic3106) and via Inter IC sound (i2s) protocol this data from Audio codec is captured in Var-Som-MX6 board for further process.

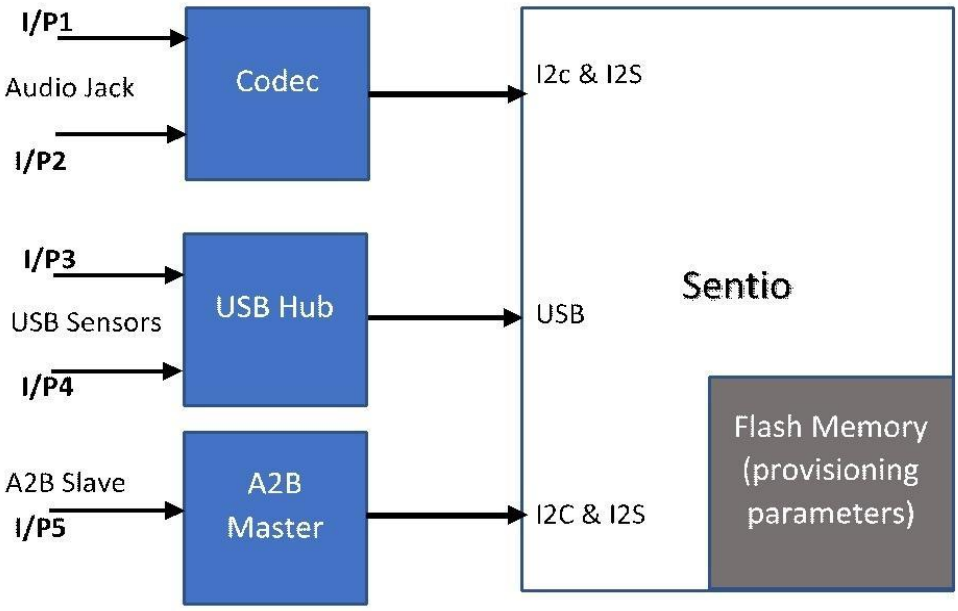


Figure 5.1
Audio Capture Architectural Model

In this audio data capturing process, through Inter IC communication (i2c) bus, I have to configure audio codec such that data can be captured through both onboard digital as well as line-in mic. For this I had to program ADC, PGA and GPIO of audio codec to get the audio data to the TLV320aic3106 audio codec that captured digital data, then transferred it to Var-Som-MX6 device and by using Advanced Linux Sound Architecture(a kernel level framework for sound card device driver),that data is abstracted.

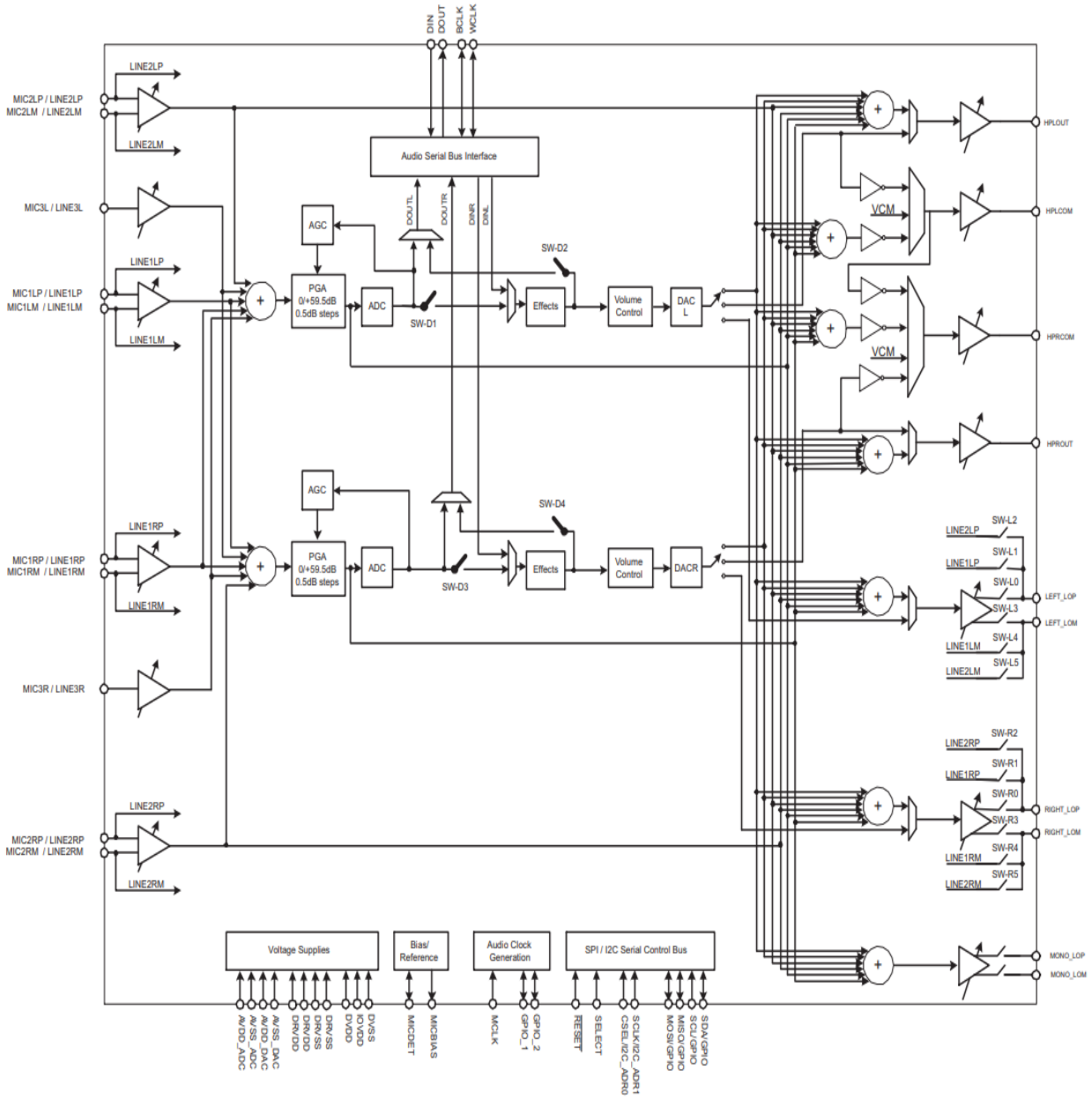


Figure 5.2
Signal Path of Codec

Line-In and digital microphone are stereo devices and USB microphone is a mono device. To get the data of both the devices, so that the data didn't get corrupt, I had to design a module which could dynamically configure and capture channel such that both stereo as well as mono audio data could be captured. Stereo data is configured in codec as Interleaved i.e. the data of left and right channel are mixed together in audio codec as one contiguous block of data and sent over to i2s bus.

This module has feature of dynamically controlling PGA i.e. Programmable Gain Amplifier and Sampling rate of audio samples. Programmable Gain Amplifier has range of 0 to 59.5db at step of 0.5db. We can configure gain of data using PGA for only Line-In because digital microphone bypasses PGA and ADC both. The TLV320AIC3106 supports the following standard audio sampling rates: 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, and 96kHz. This module configures codec to obtain audio samples at all the sampling rate which are standard to the audio codec.

This module also takes care of problem of mixing audio samples from line-in and digital microphone. When the data is captured from digital microphone, the module muted PGA gain, so data coming from line-in mic gets muted by setting bit 8 to 1 of Left and Right ADC PGA Gain Control Register and thus dies not get mixed with digital microphone data. When data was collected from the line-in mic bit 0

and 1 Audio Serial Data Interface Control Register A, which is register 08 of Audio Codec are set to 00 bits to disable digital microphone support to avoid data capturing from digital microphone.

5.2 Provisioning Module

In telecommunication, provisioning involves the process of preparing and equipping a network to allow it to provide new services to its users. Provisioning equates to "initiation" and includes altering the state of an existing priority service or capability.

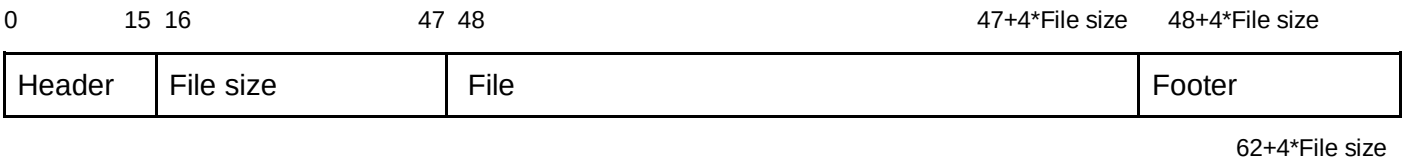
This module consists of 2 module -

- Host module
- Board module

Host module -

This module is menu driven module based on graphical user interface, which provisions selected menu parameter based on the value entered by the user of parameter like input device, SSID, password, sampling rate, etc. When the user saves the parameter by clicking save option in the menu all the changes are saved into JavaScript Object Notation (JSON) file and that json file then transferred to the board in Format given below over UART.

File Transfer Format:



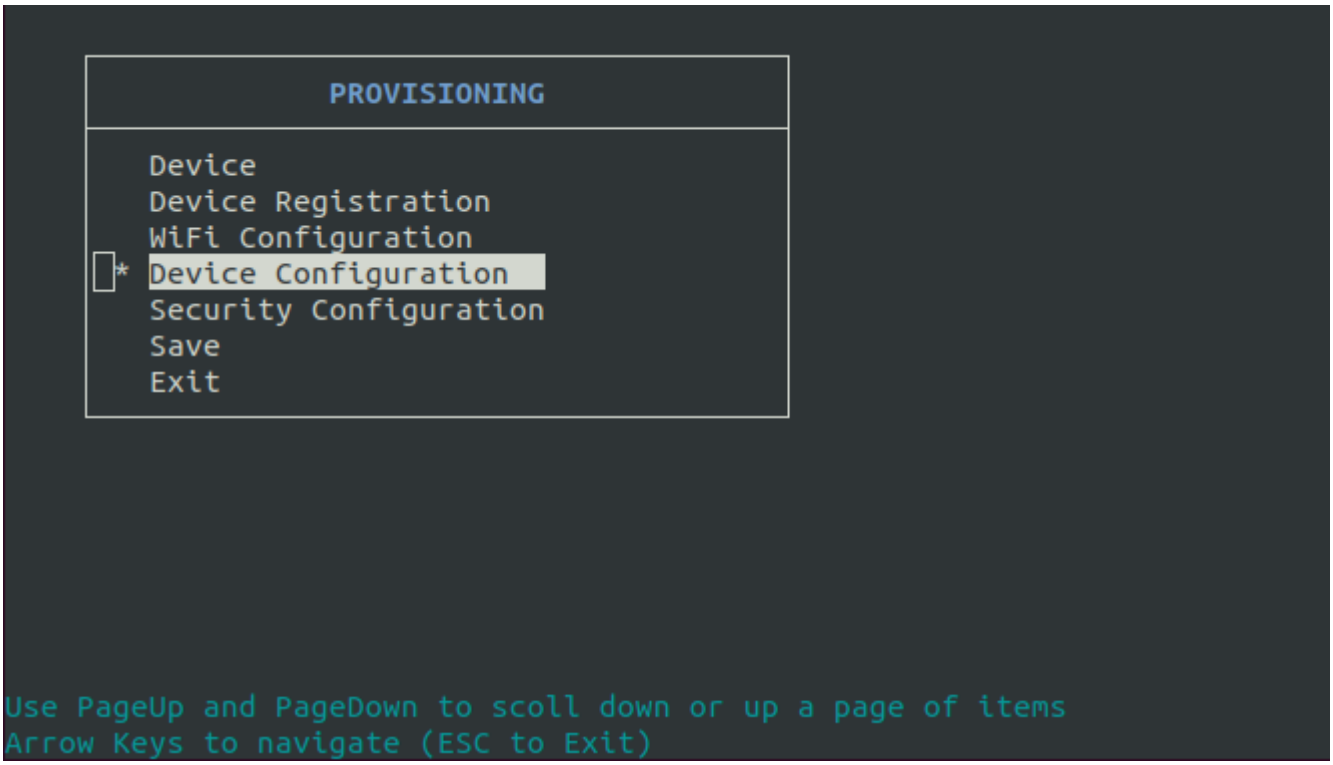


Figure 5.3
Provisioning App Interface

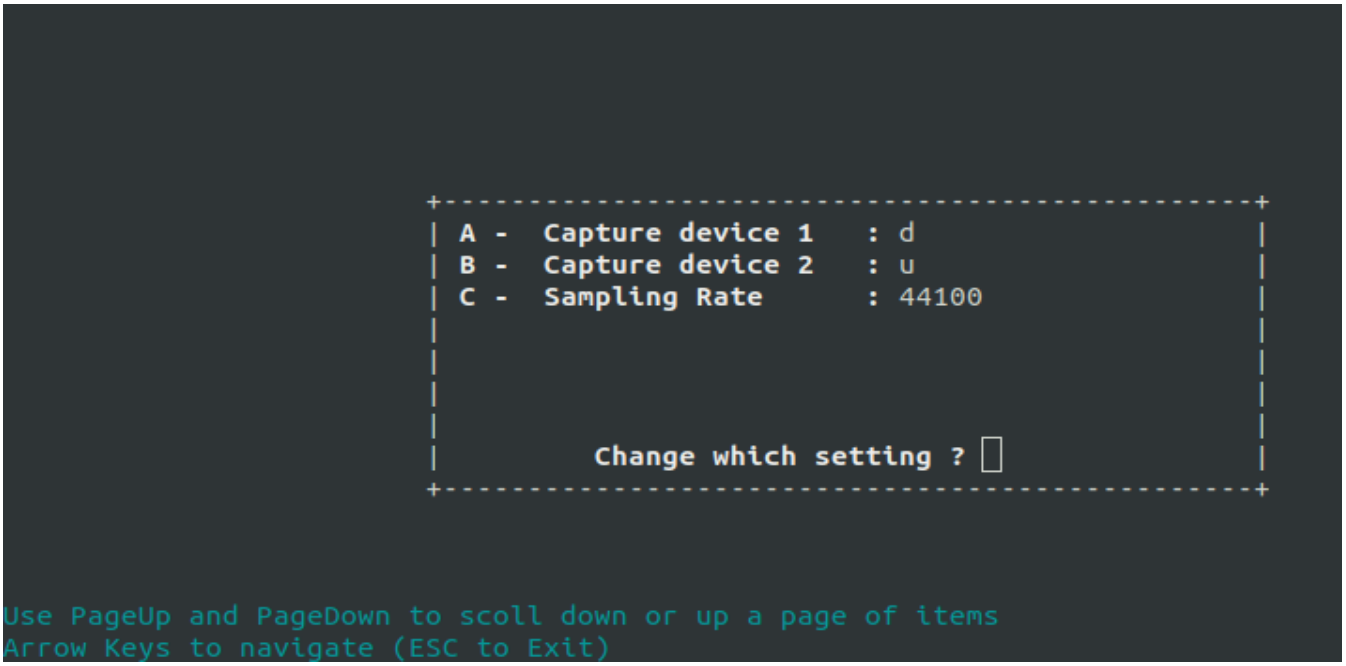


Figure 5.4
Provisioning App parameter

Board module -

This module waits at the UART 3 port to receive the header, that is of 2 bytes. When the received header matches with the programmed header, the module reads next 4 bytes that have information of the number of bytes stored in the file. Then based on the file size Var-Som-Mx6 board reads bytes based on the file size. At this module, one receives footer to check whether all the data sent by the host over UART are error free.

5.3 Power on Self Utility Test Module

This module verifies the functionality of peripheral devices and check the internet connectivity status of the i.MX6 board on powering on the board. This module checks the working of following devices:

- Line-IN mic
- Digital Mic
- USB Mic
- Wi-Fi
- Ethernet

This module has the functionality of capturing and recording acoustic sound from USB, line-in and digital microphones. This module checks whether the microphone is connected and also checks the availability of handle to capture the sound, this module checks all the functionality of all the configuration available in provisioning module i.e. whether codec the codec is able to capture sound data at the given sampling rate, PGA gain, etc.

Captured audio samples were recorded in Waveform Audio File Format which is .wav format.
It is also designed such that it will record sound, both of stereo and mono audio data i.e. for different number of channels.

Internet Connectivity of ethernet and Wi-Fi are verified using ICMP. The module checks whether the ethernet cable is connected to i.MX6 board and board is able to ping over to certain IP address or not. This module creates of RAW SOCKET and pack a packet as per standardized packet structure of ICMP and receives the packet from that IP address and unpack to obtain information like time to receive the packet, size of packet, etc. This module waits for 2s to receive the packet to create a non-blocking socket so that other functionality does not get blocked. If the packet is not received in time, this module logs an error.

In all the hardware devices ethernet is prioritized over Wi-Fi i.e. when both ethernet and Wi-Fi are connected, sending and receiving data over the internet takes place over the ethernet. So, to check Wi-Fi connectivity one has to pull out ethernet cable. This module also solves this problem by pprogrammably disabling ethernet, checking the Wi-Fi connectivity and then enabling the ethernet.

5.4 Debug Module

This module logs messages of different log level and had the functionality of storing logs into file. The module prints the operation log messages on the console terminal or in log file or both.
The logger options available in this module were:

1. Log message output: print_on_console or log_file or both
2. Logging level: NONE
   INFO
   WARN
   ERROR
   DEBUG

Log message format:

<Timestamp>: <Log Label>: message: [_FUNC_NAME_]: [_LINE_NUMBER_]

1. Timestamp: Time at which log has been generated.
   Timestamp format: yyyymmdd-hhmmss
2. Log labels
   INFO
   WARN
   ERROR
   DEBUG
3. Message: The user entered messages with relevant argument
4 _FUNC_NAME_: Function which send the log message
5. _LINE_NUMBER_: At which line of code log has been generated. Such that it becomes easy to cleanup the error occurred.

The log file name format:
sentio_log_yyyymmdd.log

This module records daily log file i.e. log of different day are stored in different log files.

Chapter 6
Conclusion

In this project we are specifically intended to implement acoustic sound monitoring system on a embedded device. The proposed fault detection method is based on several low-level features extracted from sound. The future work includes improving the method, completing the implementation of the distinct fault detection module, and implementing the system in embedded device.

6.1 Future Works

The future prospect of this module lies in improving the method, completing the implementation of the distinct fault detection module and implementing the system using accelerometer sensor along with adding temperature sensor to detect the temperature of the machine.

Also, an algorithm can also write to pinpoint the type of the fault that has occurred in the machine- may be because of the worn-out parts or the excessive usage or human error and also to itself suggest how this fault can be eradicated.

References

[1] Texas Instruments – TLV320AIC3106 Low-Power Stereo Audio CODEC for Portable Audio/Telephony Datasheet (Rev. F)
[2] Variscite – VAR-SOM-SOLO/DUAL_v1.X_Datasheet
[3] Variscite – var-solocustomboard_sch_v1
[4] A Sound Monitoring System for Fault Detection of Machine and Machining States
[5] Gong, X., and Qiao, W. Current-based mechanical fault detection for direct-drive wind turbines via synchronous sampling and impulse detection. Industrial Electronics, IEEE Transactions on 62, 3 (March 2015), 1693–1702.
[6] ALSA Framework Documentation
[7] Ncurses Documentation