ST.ANNE'S

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, New Delhi. Affiliated to Anna University, Chennai)

Accredited by NAAC

ANGUCHETTYPALAYAM, PANRUTI – 607 106.

EC3461 - COMMUNICATION SYSTEMS LABORATORY

OBSERVATION NOTE

(FOR III B.E ELECTRONICS AND COMMUNICATION ENGINEERING)

NAME : ____

REGISTER NO : ____

YEAR/SEMESTER: II Year / IV Semester

PERIOD : FEB 2025 – MAY 2025

AS PER ANNA UNIVERSITY (CHENNAI) SYLLABUS

2021 REGULATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PREPARED BY: Mr. S. BALABASKER, AP/ECE

ABOUT OBSERVATION NOTES & PREPARATION OF RECORD

- This Observation contains the basic diagrams of the circuits listed in the syllabus of the **EC3461 - COMMUNICATION SYSTEMS LABORATORY** course, along with the design of various components of the circuit and controller.
- The experiment's aim is also given at the beginning of each experiment. Once the student can design the circuit as per the circuit diagram, they are supposed to go through the instructions carefully and do the experiments step by step.
- They should note down the readings (observations) and tabulate them as specified.
- It is also expected that the students prepare the theory relevant to the experiment referring to prescribed reference books/journals in advance, and carry out the experiment after thoroughly understanding the concept and procedure.
- They should get their observations verified and signed by the staff within two days and prepare & submit the record of the experiment when they come to the laboratory in the subsequent week.
- The record should contain the experiment No., Date, Aim, Apparatus required, Theory, Procedure, and result on one side (i.e., Right-hand side, where rulings are provided) and Circuit diagram, Design, Model Graphs, Tabulations, and Calculations on the other side (i.e., Left-hand side, where no rulings are provided)
- All the diagrams and table lines should be drawn in pencil
- The students are directed to discuss & clarify their doubts with the staff members as and when required. They are also directed to follow strictly the guidelines specified.

EC3461 - COMMUNICATION SYSTEMS LABORATORY

SYLLABUS

COURSE OBJECTIVES:

- ❖ To study the AM & FM Modulation and Demodulation.
- ❖ To learn and realize the effects of sampling and TDM.
- ❖ To understand the PCM & Digital Modulation.
- ❖ To Simulate Digital Modulation Schemes.
- ❖ To Implement Equalization Algorithms and Error Control Coding Schemes.

LIST OF EXPERIMENTS

1. AM- Modulator and Demodulator
2. FM - Modulator and Demodulator
3. Pre-Emphasis and De-Emphasis.
4. Signal sampling and TDM.
5. Pulse Code Modulation and Demodulation.
6. Pulse Amplitude Modulation and Demodulation.
7. Pulse Position Modulation and Demodulation and Pulse Width Modulation and Demodulation.
8. Digital Modulation – ASK, PSK, FSK.
9. Delta Modulation and Demodulation.
10. Simulation of ASK, FSK, and BPSK Generation and Detection Schemes.
11. Simulation of DPSK, QPSK and QAM Generation and Detection Schemes.
12. Simulation of Linear Block and Cyclic Error Control coding Schemes.

OUTCOMES:

CO1: Design AM, FM & Digital Modulators for specific applications.
CO2: Compute the sampling frequency for digital modulation.
CO3: Simulate & validate the various functional modules of Communication system.
CO4: Demonstrate their knowledge in base band signaling schemes through implementation of digital modulation schemes.
CO5: Apply various channel coding schemes & demonstrate their capabilities towards the improvement of the noise performance of Communication system.

LIST OF EXPERIMENTS

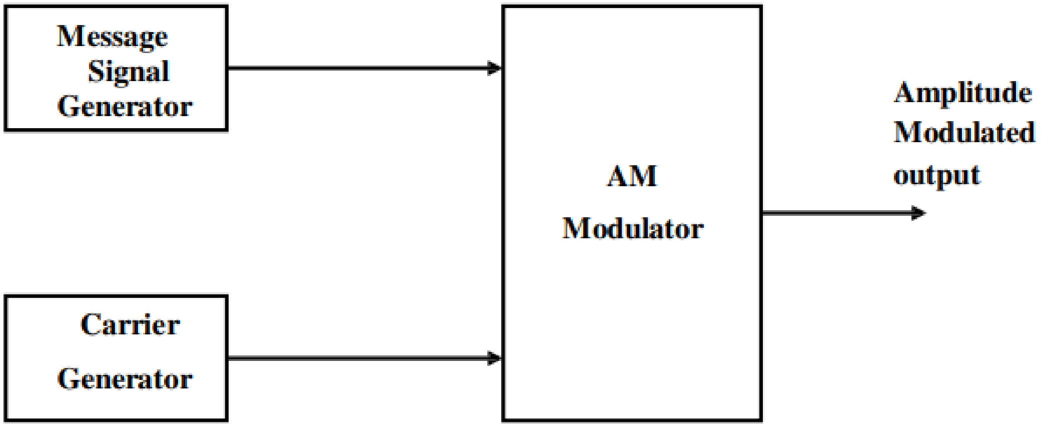
S.No.	DATE	NAME OF THE EXPERIMENT	PAGE NO	DATE OF SUBMISSION	MARK (10)	SIGNATURE

LIST OF EXPERIMENTS

S.No.	DATE	NAME OF THE EXPERIMENT	PAGE NO	DATE OF SUBMISSION	MARK (10)	SIGNATURE

BLOCK DIAGRAM

AM MODULATOR:



AM DEMODULATOR:

EXP NO:	AM- Modulator and Demodulator
DATE	

AIM:

To study and analyze the Amplitude Modulation (AM) and Demodulation process using the AM Transmitter & Receiver Trainer Kit, and to observe the modulated and demodulated signals.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	AM Transmitter Kit	1
2	AM Receiver Kit	1
3	DSO	1
4	Probe	Few
5	Patch cord	Few

THEORY:

AM Modulation:

Amplitude Modulation (AM) is a technique in which the amplitude of a high-frequency carrier wave is varied in proportion to the instantaneous value of the message signal. The VI Microsystems Trainer Kit provides a hands-on platform for implementing and analyzing AM modulation and demodulation processes.

1. Amplitude Modulation (AM) Process

- Modulation Principle: The message signal m(t) is superimposed on a carrier signal c(t) by varying its amplitude.
- Mathematical Representation: $s(t) = [A + m(t)] \cos(2\pi f_c t)$

where:

- A is the carrier amplitude,
- m(t) is the message signal,
- f_c is the carrier frequency,
- s(t) is the modulated signal.

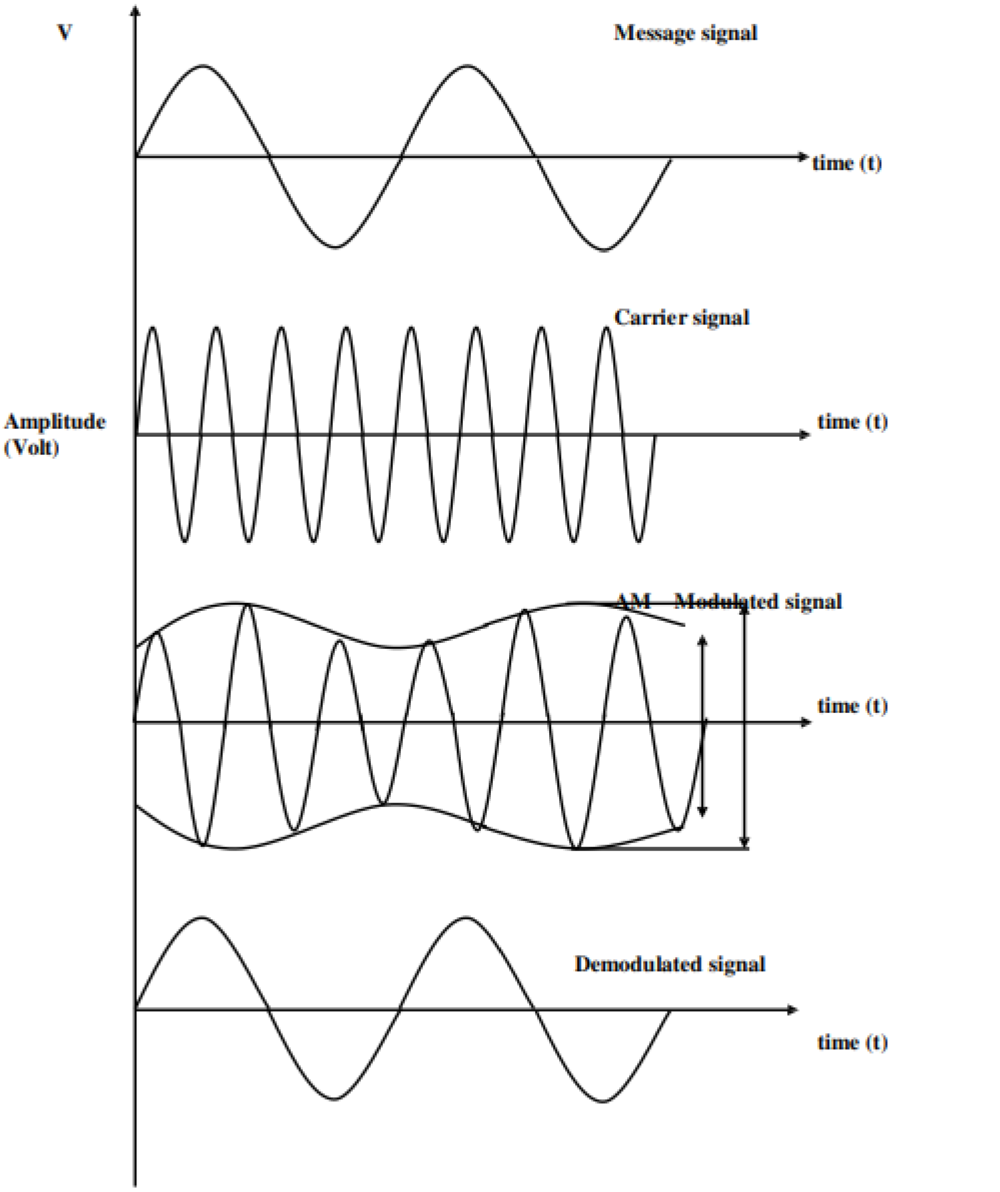
- Types of AM:
  - **Double-Sideband Full Carrier (DSB-FC)** – The carrier and both sidebands are transmitted.
  - **Double-Sideband Suppressed Carrier (DSB-SC)** – Only sidebands are transmitted, reducing power consumption.
  - **Single-Sideband (SSB)** – Only one sideband is transmitted, improving bandwidth efficiency.

2. **Amplitude Demodulation Process**

Demodulation is the process of recovering the original message signal from the modulated AM signal. The VI Microsystems Trainer Kit provides hardware components for AM detection using:

- Envelope Detector: Detects the envelope of the AM signal, providing an approximation of the original message signal.
- Coherent Detection: Uses a synchronized carrier signal to extract the original message signal.

MODEL GRAPH: -


V
Message signal
time (t)
Carrier signal
Amplitude
(Volt)
time (t)
AM Modulated signal
time (t)
Demodulated signal
time (t)

TABULATION:

TRANSMITTER

Signal	Amplitude	Time
Modulating Signal		
Carrier Signal		
Amplitude Modulated Wave		

RECEIVER:

Signal	Amplitude	Time
Amplitude Modulated Wave		
Demodulated Signal		

PROCEDURE:

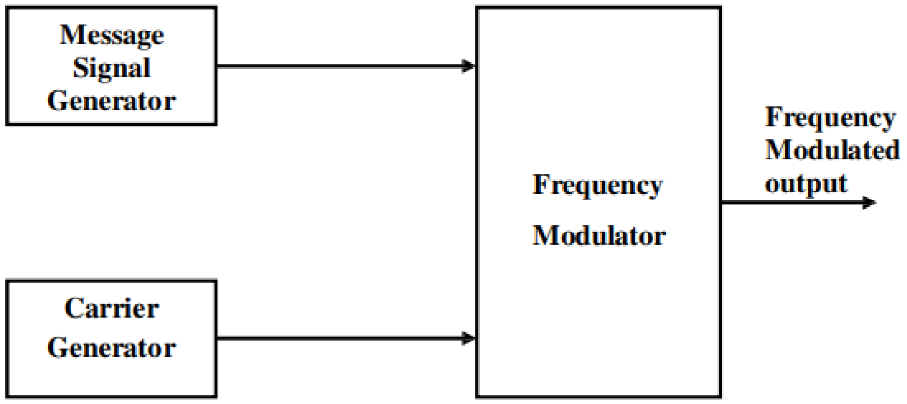
1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to the AM modulator.
3. High frequency message signal is given as one input to the AM modulator.
4. The amplitude-modulated waveform obtained is viewed in the CRO.
5. Readings are taken for message, carrier, and amplitude-modulated wave.
6. The modulated wave is given as input to the envelope detector.
7. The demodulated output is noted in the CRO.
8. Modulation index has to be calculated as per the formula.

Modulation Index $=\frac{Emax - Emin}{Emax + Emin}$

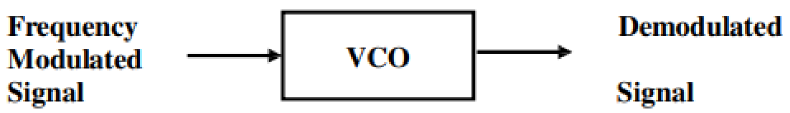
RESULT:

BLOCK DIAGRAM

FM MODULATOR:



FM DEMODULATOR:



EXP NO:	FM- Modulator and Demodulator
DATE	

AIM:

To study and analyze the Frequency Modulation (FM) and Demodulation process using the FM Transmitter & Receiver Kit and to observe the modulated and demodulated signals.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	FM Transmitter Kit	1
2	FM Receiver Kit	1
3	DSO	1
4	Probe	Few
5	Patch cord	Few

THEORY:

Frequency Modulation (FM) is a modulation technique in which the frequency of the carrier signal is varied in accordance with the instantaneous amplitude of the message signal while keeping its amplitude constant. FM is widely used in radio broadcasting, communication systems, and radar applications.

1. Frequency Modulation (FM) Process

- Modulation Principle: The frequency of the carrier signal is varied based on the amplitude of the message signal.
- Mathematical Representation: $s(t) = A \cos[(2\pi f_c t) + 2\pi k_f \int m(t)dt]$

where:

- A is the carrier amplitude,
- f_c is the carrier frequency,
- k_f is the frequency sensitivity,
- m(t) is the message signal,
- s(t) is the modulated signal.

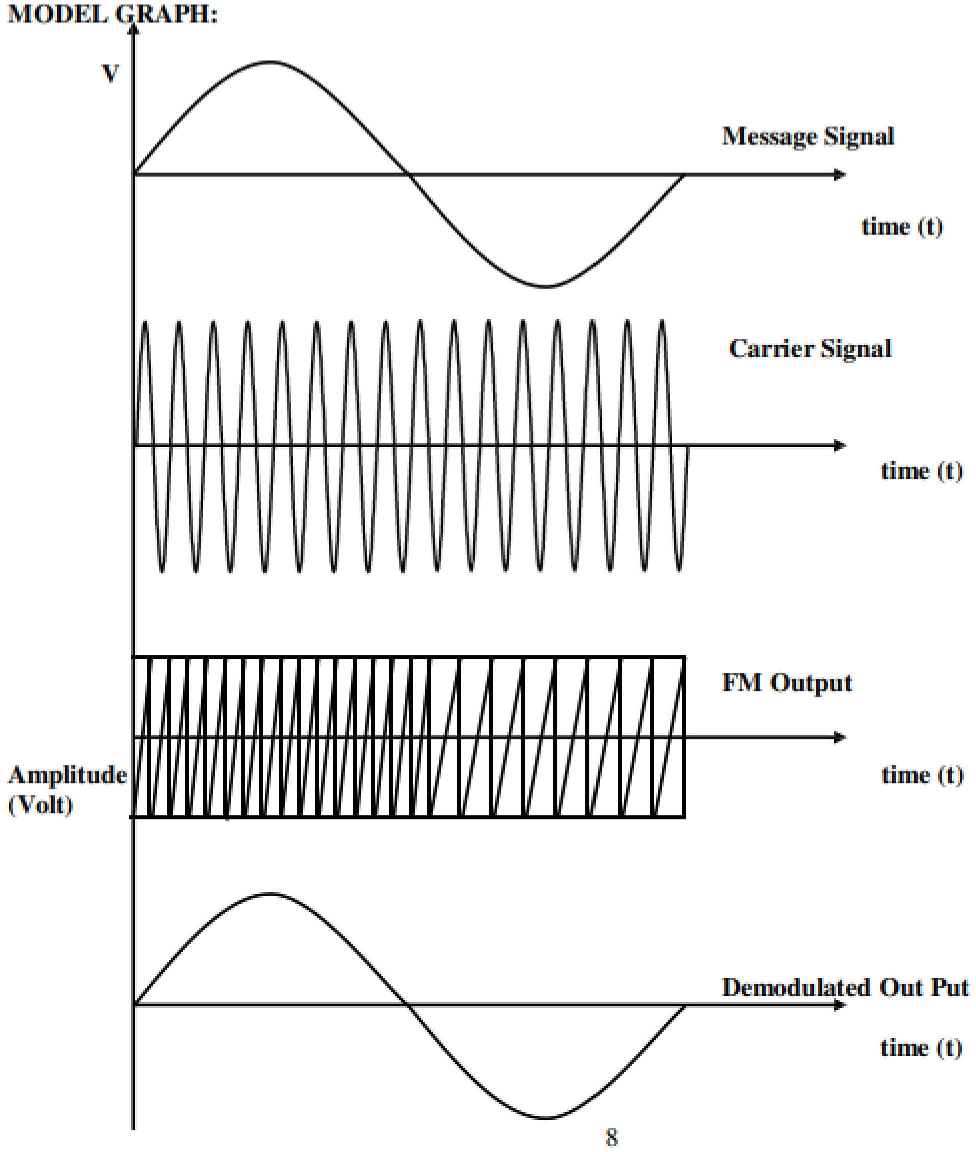
- Types of FM:
    - Narrowband FM (NBFM) – Used for communication systems with limited bandwidth.
    - Wideband FM (WBFM) – Used for high-fidelity audio transmission, such as FM radio broadcasting.

2. Frequency Demodulation Process

Demodulation is the process of retrieving the original message signal from the FM signal. The VI Microsystems Trainer Kit includes components for FM demodulation using:

- Slope Detector: Converts frequency variations into amplitude variations, which are then detected using an envelope detector.
- Phase-Locked Loop (PLL): A feedback system that locks onto the carrier frequency and extracts the message signal.

MODEL GRAPH:


V
Message Signal
time (t)
Carrier Signal
time (t)
FM Output
Amplitude
(Volt)
time (t)
Demodulated Out Put
time (t)
8

TABULATION:

TRANSMITTER

Signal	Amplitude	Time

RECEIVER:

Signal	Amplitude	Time

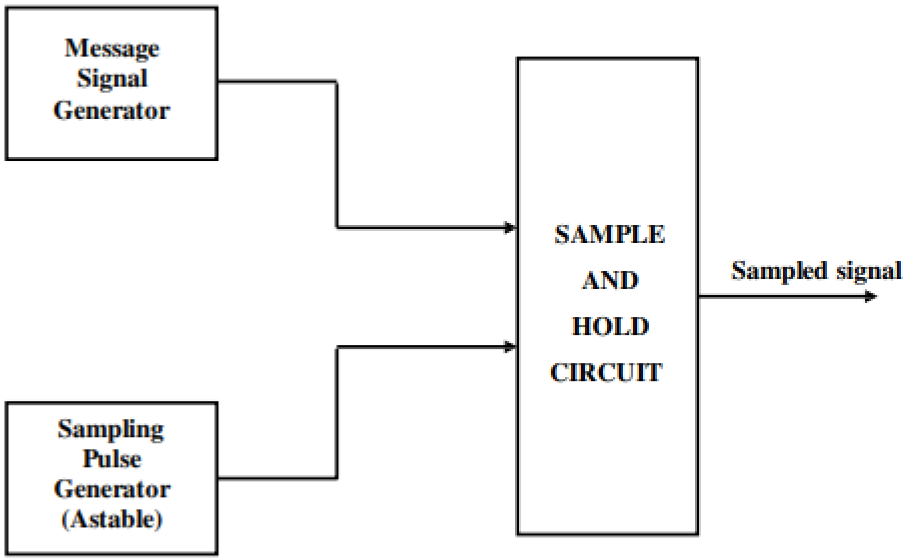
PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to FM modulator.
3. High frequency message signal is given as one input to FM modulator.
4. The amplitude modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and amplitude modulated wave.
6. The modulated wave is given as input to envelope detector.
7. The demodulated output is noted in CRO.
8. Modulation index have to be calculated as per formula.

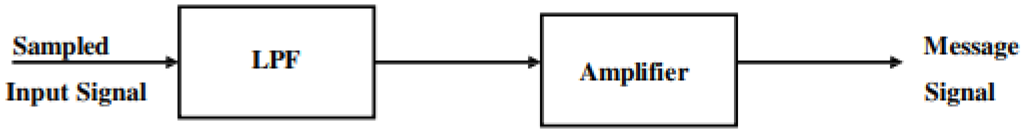
RESULT:

BLOCK DIAGRAM

SAMPLER



RECONSTURCTION CIRCUIT



EXP NO:	SIGNAL SAMPLING AND RECONSTRUCTION
DATE	

AIM:

To study and analyze the process of signal sampling and reconstruction, including natural sampling, flat-top sampling, and sample-and-hold techniques, using Analog signal sampler kit.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	Analog signal sampler kit	1
2	Reconstruction Kit	1
3	DSO	1
4	Probe	Few
5	Patch cord	Few

THEORY:

Sampling is the process of converting a continuous-time signal into a discrete-time signal by taking periodic samples of the signal at a fixed rate. The Nyquist theorem states that a signal can be completely reconstructed if it is sampled at a rate at least twice the highest frequency present in the signal.

1. Types of Sampling Techniques

A. Natural Sampling

- In natural sampling, the message signal is multiplied by a periodic pulse train, but the amplitude of the samples follows the original signal's variations.
- This technique retains the shape of the original waveform within each pulse duration.
- Mathematical Representation: where:
    - is the message signal,
    - is a periodic pulse train,
    - is the sampled signal.

B. Flat-Top Sampling

- In flat-top sampling, the samples are held at a constant amplitude for the duration of the sampling pulse.
- This introduces a distortion called aperture effect, which can be minimized using appropriate reconstruction filters.
- Mathematical Representation: for in the pulse duration, where represents the discrete sampling instances.

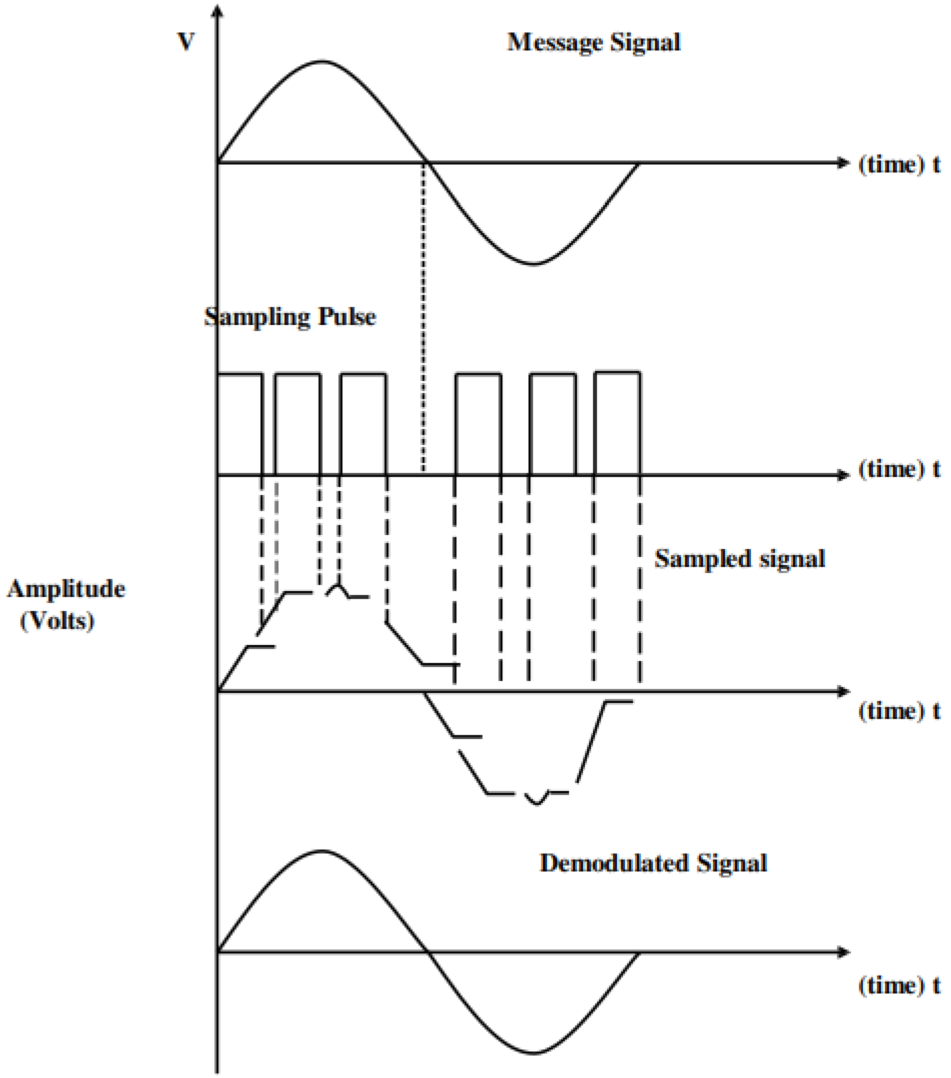
C. Sample-and-Hold Technique

- This method holds each sample for a fixed duration before updating to the next sample.
- It is commonly used in analog-to-digital converters (ADCs) to maintain a stable input for processing.
- Sample-and-hold circuits prevent high-frequency variations in the signal from causing inaccuracies in conversion.

2. Signal Reconstruction

- Reconstruction is the process of recovering the original continuous signal from its discrete samples.
- This is typically achieved using a low-pass filter to remove high-frequency components introduced during sampling.
- The ideal reconstruction filter is a sinc function, but practical filters such as Butterworth or Chebyshev filters are used in real applications.

MODEL GRAPH


V
Message Signal
(time) t
Sampling Pulse
(time) t
Sampled signal
Amplitude
(Volts)
(time) t
Demodulated Signal
(time) t

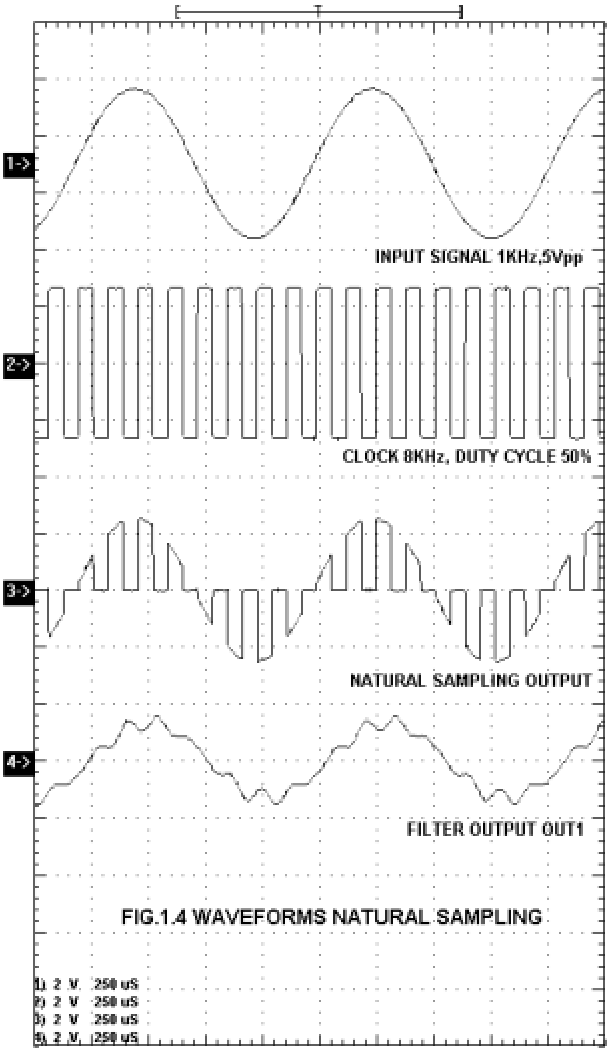


FIG.1.4 WAVEFORMS NATURAL SAMPLING

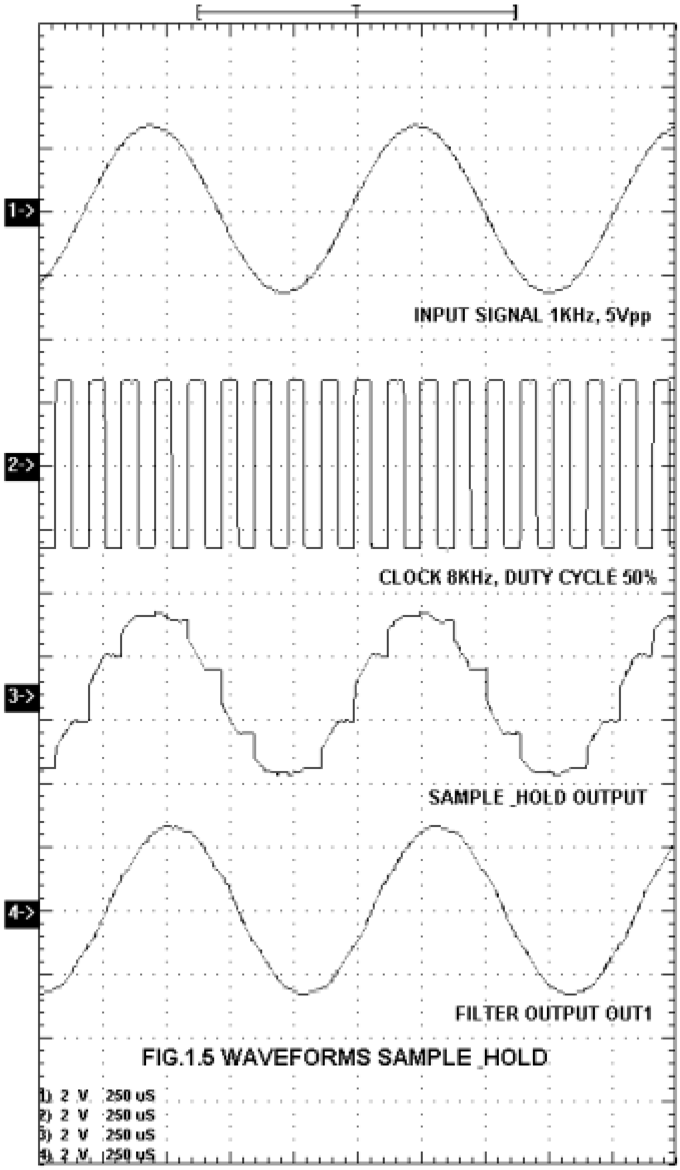


FIG.1.5 WAVEFORMS SAMPLE HOLD

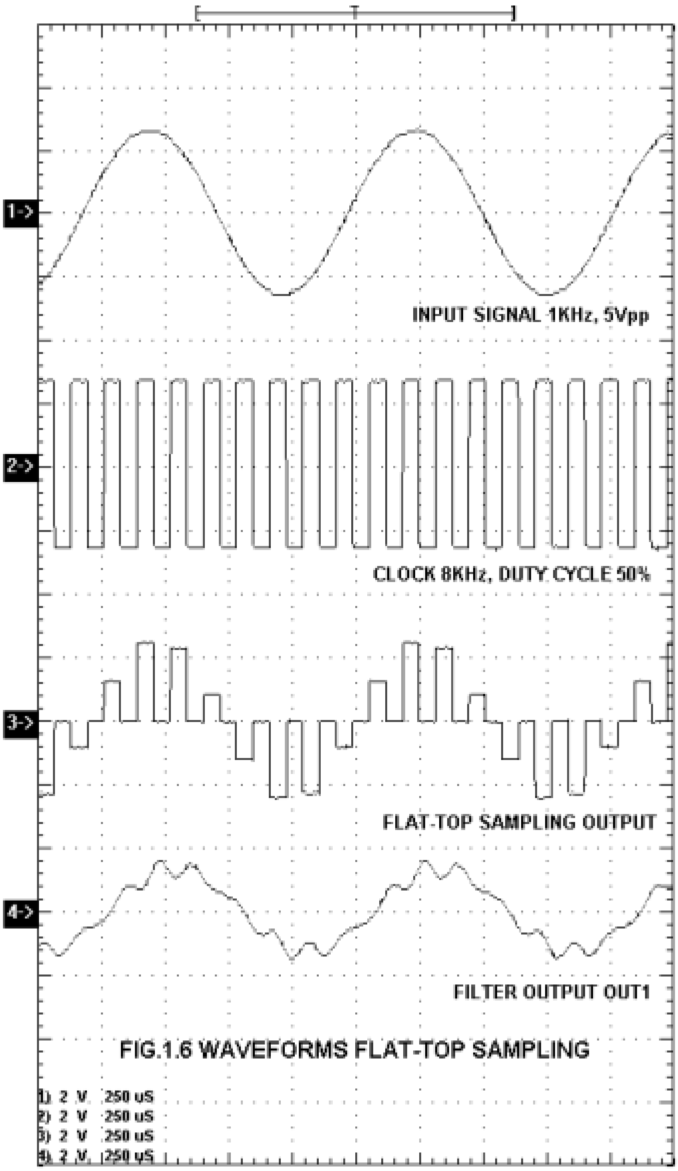


FIG.1.6 WAVEFORMS FLAT-TOP SAMPLING

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to sample/hold circuit
3. Carrier pulse signal is given as another input to to sample/hold circuit.
4. The sampling pulse waveform obtained is viewed in the CRO.
5. Readings are taken for message, carrier, and sampling pulse.
6. The sampled wave is given as input to the reconstruction circuit.
7. The reconstructed circuit output is noted in CRO

TABULATION:

TRANSMITTER

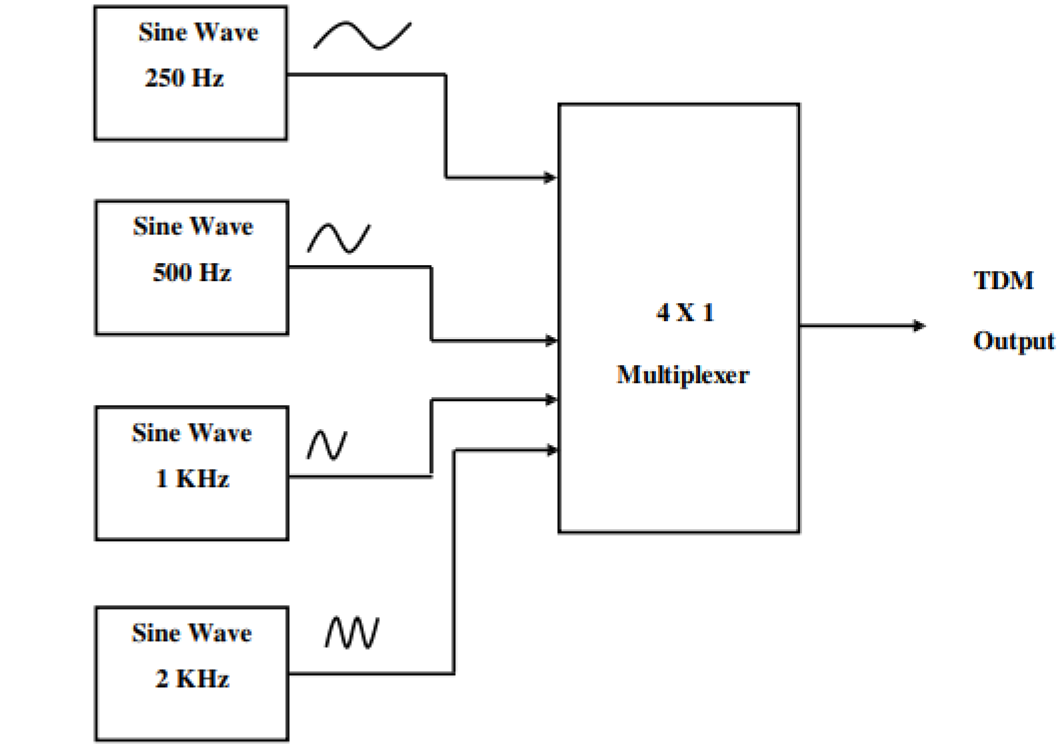
Signal	Amplitude	Time

RECEIVER:

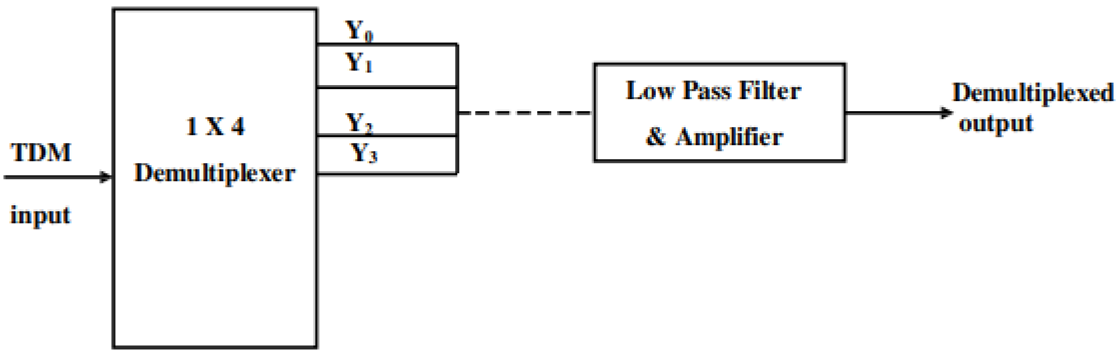
Signal	Amplitude	Time

RESULT:

TIME DIVISION MULTIPLEXER



TDM DEMULTIPLEXER



EXP NO:	TIME DIVISION MULTIPLEXING (TDM)
DATE	

AIM:

To study and understand the working of Time Division Multiplexing (TDM) using a TDM kit

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	Time Division Multiplexing Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Time Division Multiplexing (TDM) is a digital multiplexing technique in which multiple signals share the same transmission channel by dividing the time into discrete time slots. Each signal is assigned a specific time slot, ensuring that multiple signals can be transmitted over the same medium without interference.

Types of TDM

Synchronous TDM

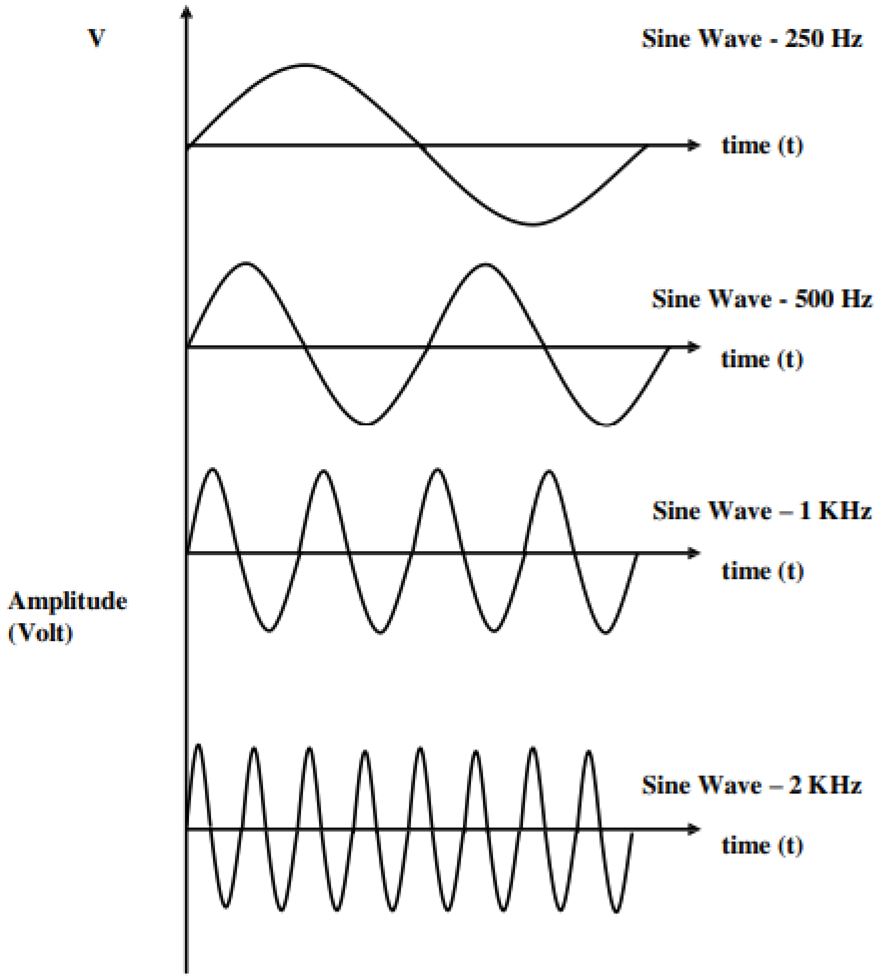
- Each source is assigned a fixed time slot, even if it has no data to transmit.
- Time slots are pre-determined and occur at regular intervals.
- Used in digital telephony (e.g., T1 and E1 lines).

Asynchronous (or Statistical) TDM

- Time slots are dynamically allocated based on demand.
- More efficient use of bandwidth.
- Used in packet-switched networks.

MODEL GRAPH:-

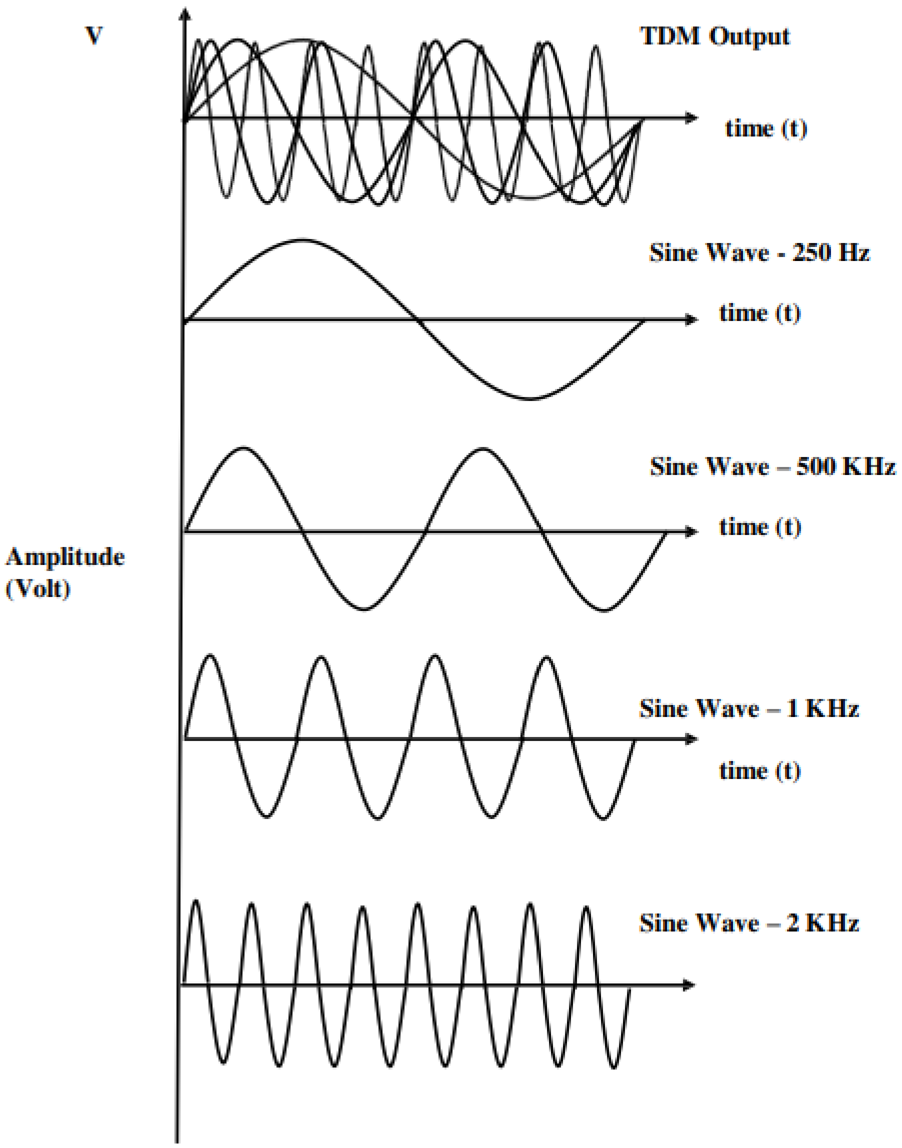
TDM INPUT



PROCEDURE:

1. Connections must be given as per the diagram.
2. Four different frequency message signals are given as input to the TDM amplitude.
3. The multiplexed waveform obtained is viewed in the CRO.
4. Readings are taken for each message signal.
5. The multiplexed wave is given as input to the demultiplexer circuit.
6. The demultiplexed output is noted in CRO.
7. The graph is plotted for multiplexed signal and, demultiplexed signal.

TDM OUTPUT


V
TDM Output
time (t)
Sine Wave - 250 Hz
time (t)
Sine Wave – 500 KHz
time (t)
Amplitude
(Volt)
Sine Wave – 1 KHz
time (t)
Sine Wave – 2 KHz

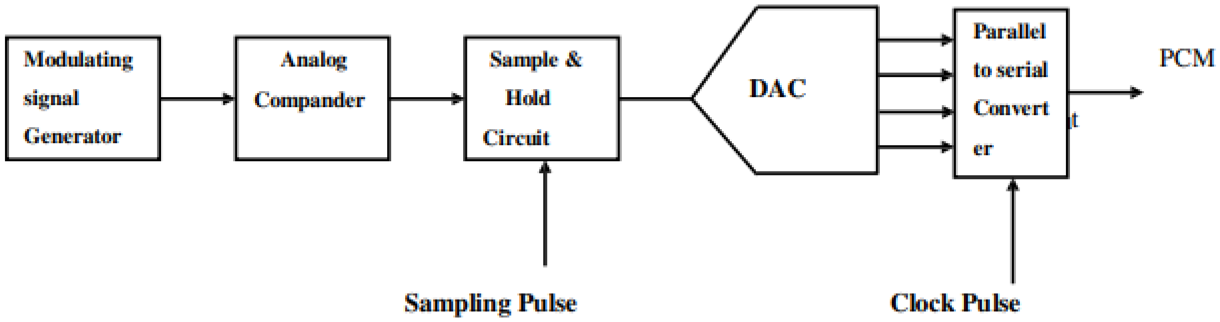
TABULATION:

Signal	Amplitude	Time
Transmitter section: Sine wave (250 Hz) Sine wave (500 Hz) Sine wave (1K Hz) Sine wave (2 KHz)		
TDM wave (Composite of all above 4 signals)		
Receiver section: Sine wave (250 Hz) Sine wave (500 Hz) Sine wave (1K Hz) Sine wave (2 KHz)		

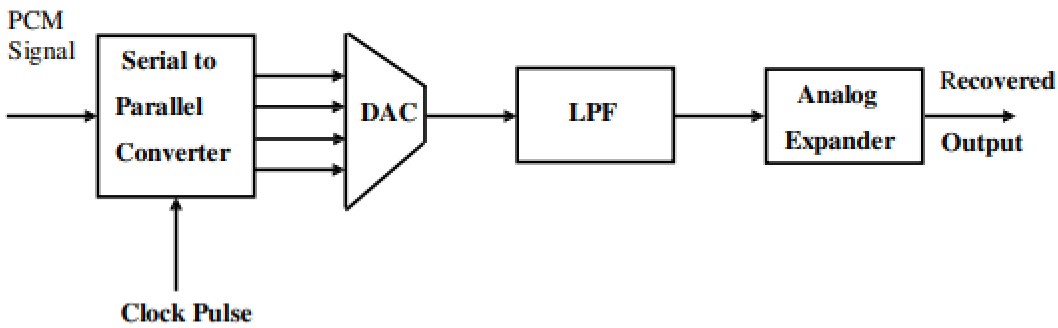
RESULT:

BLOCK DIAGRAM

PCM – PULSE CODE MODULATOR:-



PCM – DEMODULATOR:-



EXP NO:	Pulse Code Modulation and Demodulation
DATE	

AIM:

To obtain the PCM – Modulated and Demodulated signal for given message signal..

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	PCM –Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

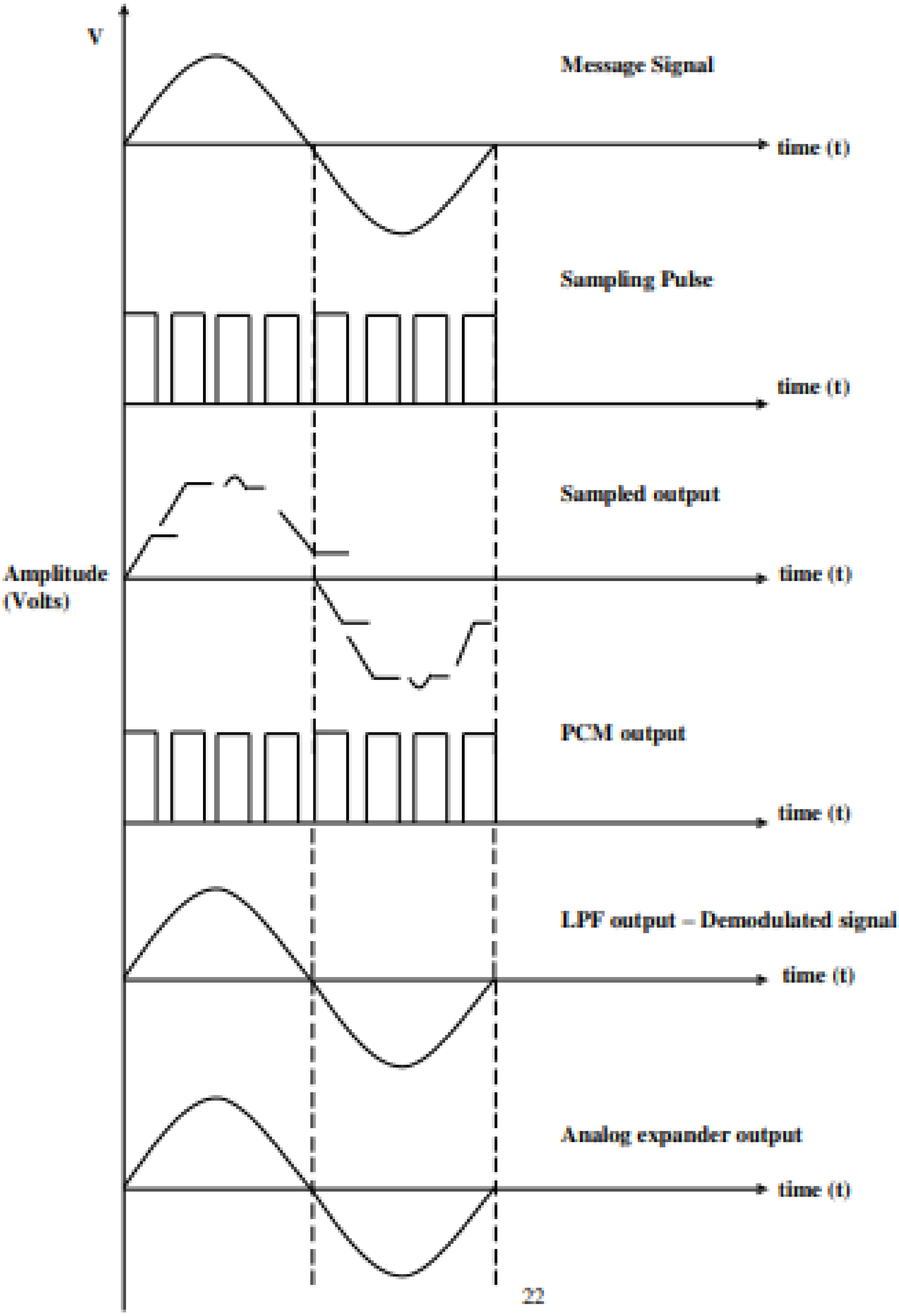
In PCM, a message signal is represented by the sequence of coded pulses. So the signal is discrete in both time and amplitude.

The basic operations performed in the PCM system are

(i) Sampling
(ii) Quantizing
(iii) Encoding

The incoming message signal is sampled with a train of rectangular pulses. To ensure perfect reconstruction at the receiver, a sampling rate fs > 2w is used. The rounding off sampled signal is called quantization. So that quantized signal is discrete in both time and amplitude. Then, the quantized signal is translated into a more appropriate form of code format by encoding. In the channel regenerators are used to increase the immunity of signal against noise. The receiver has to regenerate, reshape the received pulses, and then regroup them into a recovered signal

MODEL GRAPH:-


V
Message Signal
time (t)
Sampling Pulse
time (t)
Sampled output
Amplitude
(Volts)
time (t)
PCM output
time (t)
LPF output – Demodulated signal
time (t)
Analog expander output
time (t)
22

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to the sampler.
3. The clock pulse signal is given as another input to the sampler.
4. The sampled, encoded waveform obtained is viewed in the CRO
5. Readings are taken for message, pulse, and PCM wave.
6. The PCM wave is given as input to the demodulator circuit.
7. The demodulated. The output is noted in CRO.
8. The graph is plotted for PCM, modulated, and demodulated wave

TABULATION:

MODULATION

Signal	Amplitude	Time

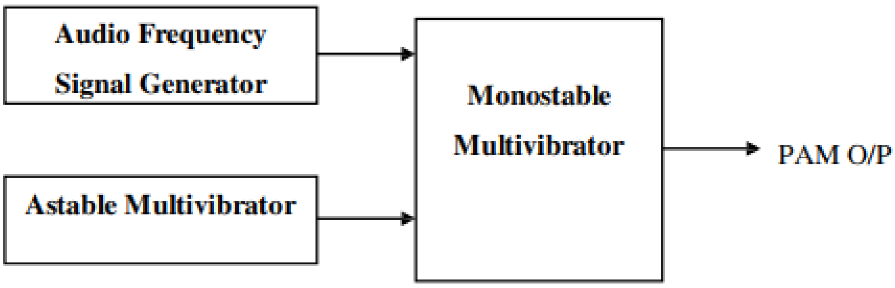
DEMODULATION:

Signal	Amplitude	Time

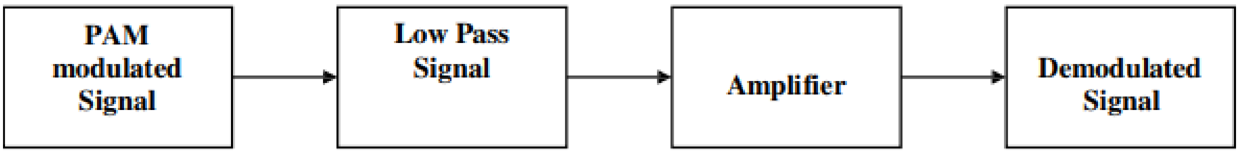
RESULT:

BLOCK DIAGRAM

PULSE AMPLITUDE MODULATOR



DEMODULATOR



EXP NO:	PULSE AMPLITUDE MODULATION
DATE	

AIM:
To study and understand the working of **Pulse Amplitude Modulation (PAM)** using a **PAM kit**

APPARATUS REQUIRED:

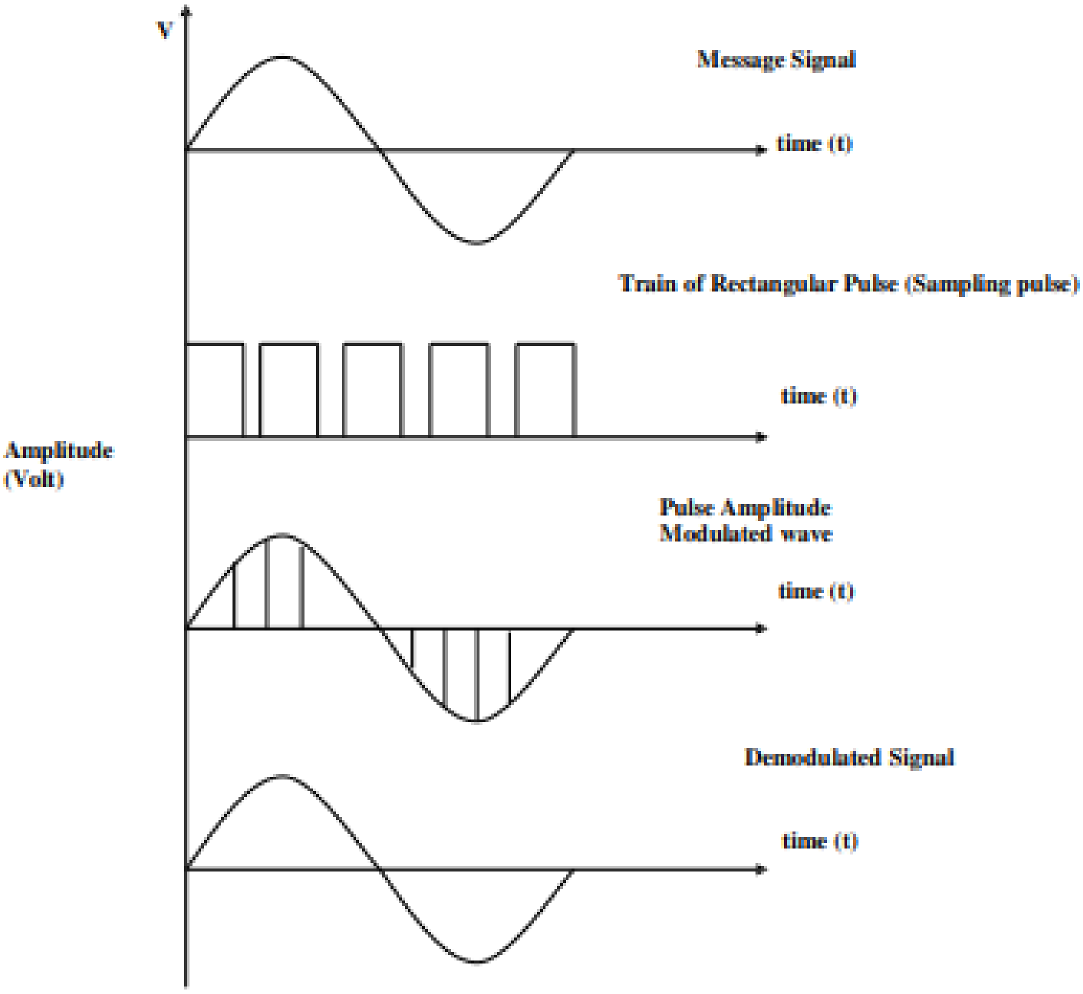
S.No	Hardware Requirements	Quantity
1	PAM Modulation and Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Pulse Amplitude Modulation (PAM) is a modulation technique where the amplitude of a series of pulses varies according to the instantaneous value of the modulating signal. There are two main types of sampling methods used in PAM:

1. **Natural Sampling**
   - In natural sampling, the pulses follow the shape of the original modulating signal during the sampling period.
   - The top of the pulses is not flat, meaning they retain the continuous variations of the signal during sampling.
   - This type of sampling reduces distortion but makes demodulation more complex.
2. **Flat-Top Sampling**
   - In flat-top sampling, the amplitude of each pulse is held constant at the value of the modulating signal at the instant of sampling.
   - The sampled pulses have a flat top, eliminating variations within the sampling interval.
   - This technique simplifies demodulation but introduces aperture distortion due to the holding process.

MODEL GRAPH:


V
Message Signal
time (t)
Train of Rectangular Pulse (Sampling pulse)
time (t)
Amplitude
(Volt)
Pulse Amplitude
Modulated wave
time (t)
Demodulated Signal
time (t)

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to PAM modulator.
3. Carrier pulse signal is given as another input to PAM modulator.
4. The pulse amplitude modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and pulse amplitude modulated wave.
6. The modulated wave is given as input to demodulator
7. The demodulated output is noted in CRO.
8. The graph is plotted for PAM

TABULATION:

MODULATION

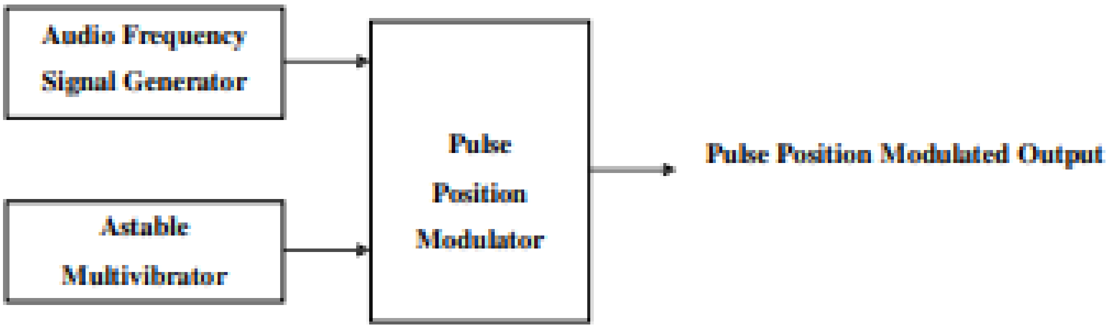
Signal	Amplitude	Time

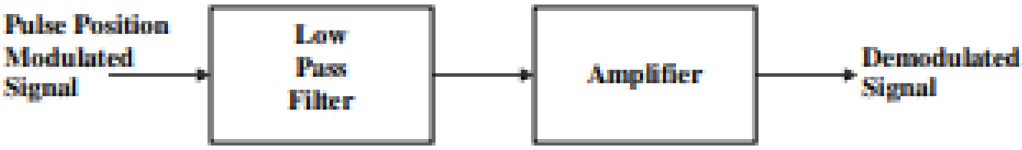
DEMODULATION

Signal	Amplitude	Time

RESULT:

BLOCK DIAGRAM





EXP NO:	PULSE POSITION MODULATION
DATE	

AIM:

To study and understand the working of **Pulse Position Modulation (PPM)** using a PPM kit

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	PPM Modulation and Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

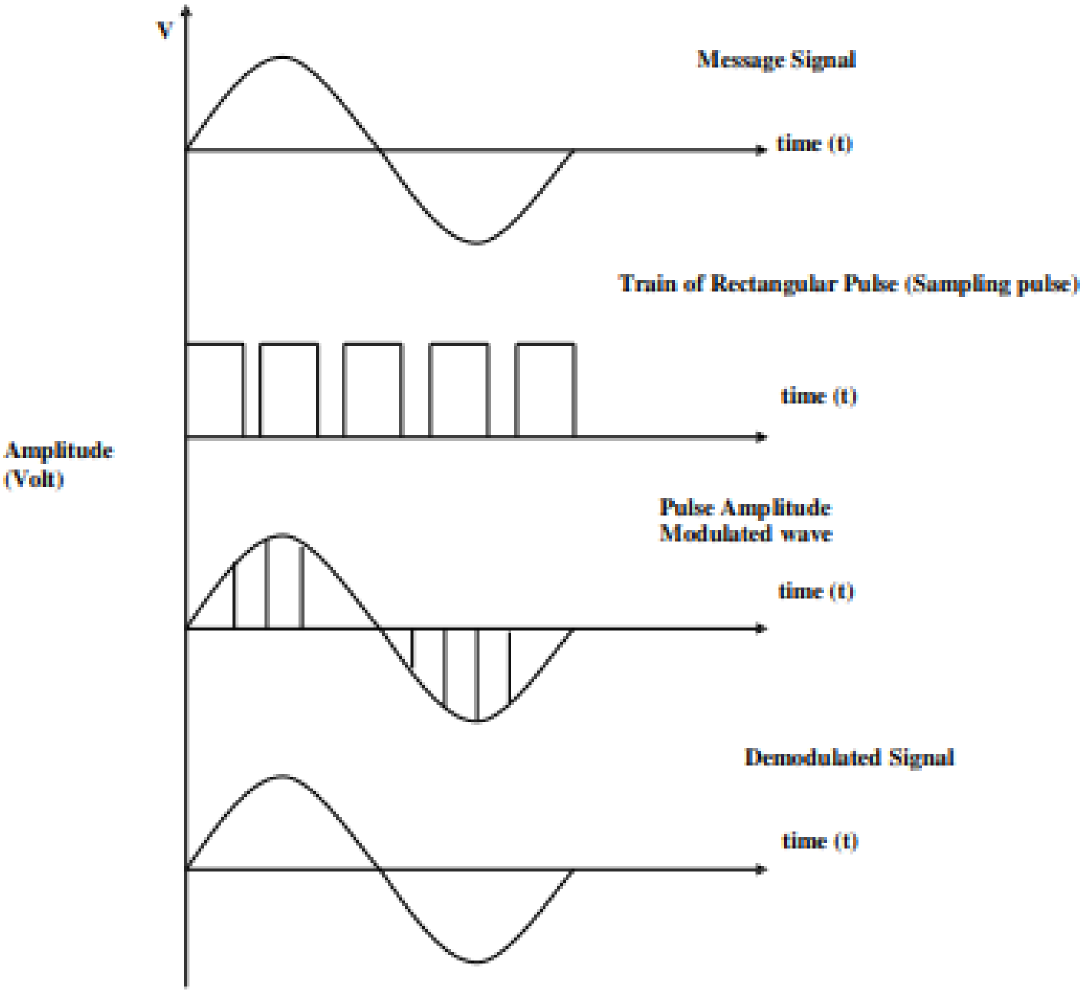
THEORY:

Pulse Position Modulation (PPM) is a type of pulse modulation where the position of a pulse is varied according to the amplitude of the modulating signal, while the width and amplitude of the pulse remain constant.

Working Principle

- The modulating signal is sampled, and its amplitude is mapped to a time delay in pulse position.
- The carrier pulses are transmitted at varying positions based on the input signal's amplitude.
- At the receiver, the original signal is reconstructed by detecting pulse positions.

MODEL GRAPH:


V
Amplitude
(Volt)
Message Signal
time (t)
Train of Rectangular Pulse (Sampling pulse)
time (t)
Pulse Amplitude
Modulated wave
time (t)
Demodulated Signal
time (t)

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to PPM modulator.
3. Carrier pulse signal is given as another input to PPM modulator.
4. The pulse amplitude modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and pulse position modulated wave.
6. The modulated wave is given as input to demodulator
7. The demodulated output is noted in CRO.
8. The graph is plotted for PPM.

TABULATION:

MODULATION

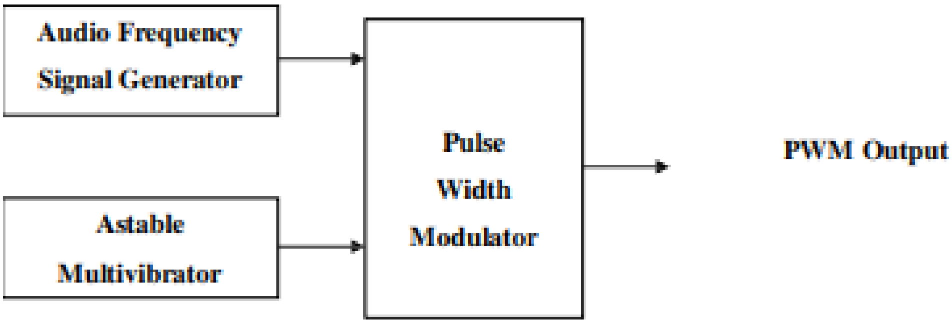
Signal	Amplitude	Time

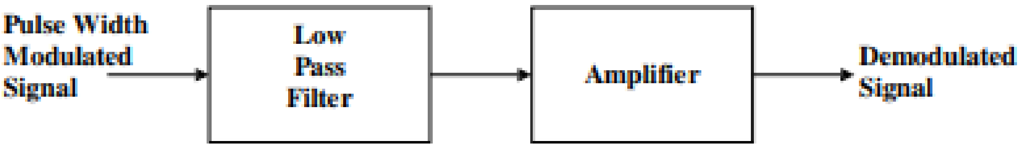
DEMODULATION

Signal	Amplitude	Time

RESULT:

BLOCK DIAGRAM





EXP NO:	PULSE WIDTH MODULATION
DATE	

AIM:
To study and understand the working of **Pulse Width Modulation (PWM)** using a PWM kit

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	PWM Modulation and Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

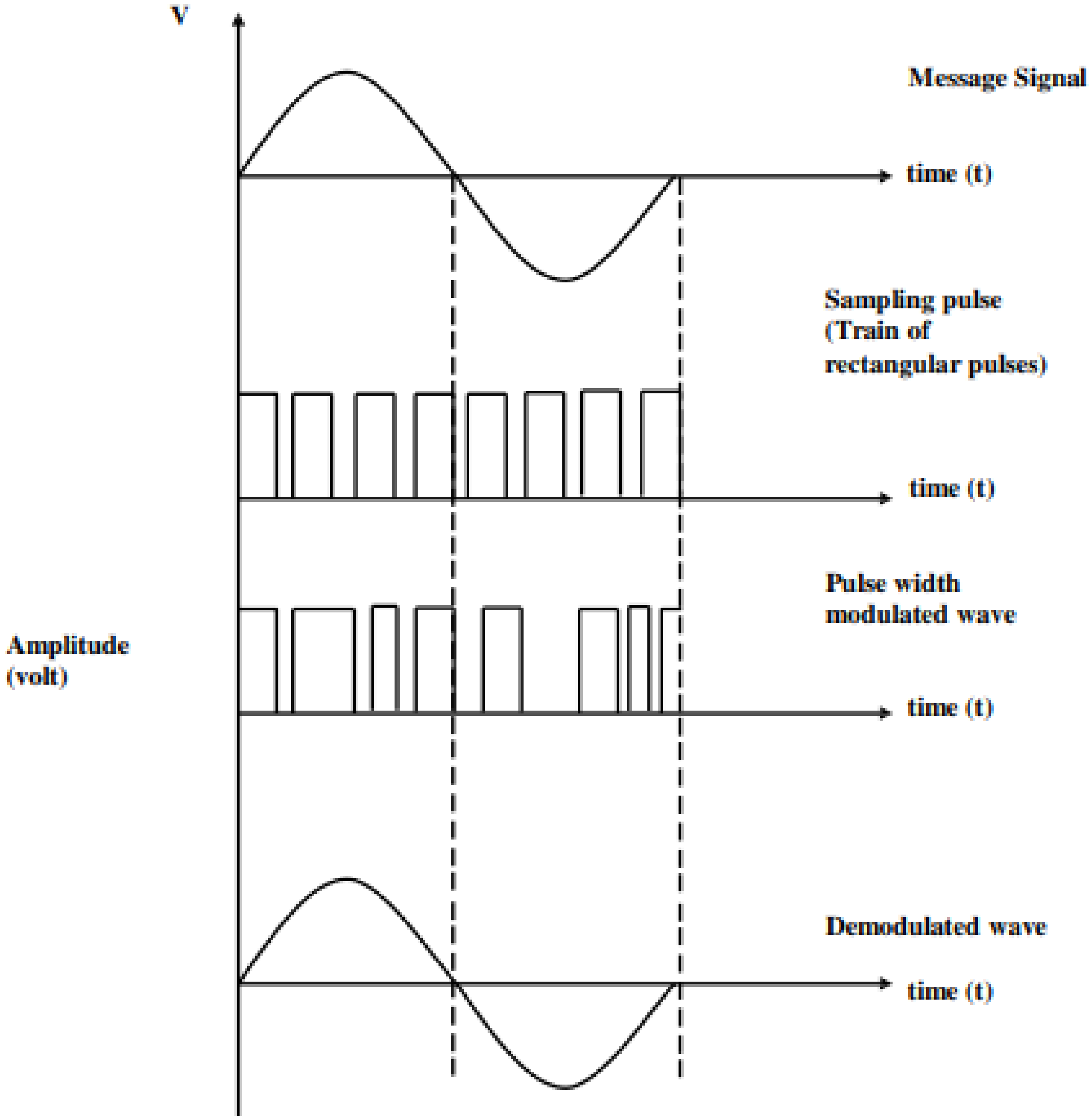
THEORY:

Pulse Width Modulation (PWM) is a modulation technique where the width (or duration) of pulses in a periodic pulse train is varied in proportion to the instantaneous amplitude of the modulating signal. The frequency and amplitude of the pulses remain constant, but the duty cycle changes according to the input signal.

Working Principle

- A continuous analog signal is sampled at regular intervals.
- The width of each pulse is adjusted based on the amplitude of the modulating signal at that instant.
- The resulting PWM signal is transmitted and later demodulated to reconstruct the original signal

MODEL GRAPH:


V
Amplitude
(volt)
Message Signal
time (t)
Sampling pulse
(Train of
rectangular pulses)
time (t)
Pulse width
modulated wave
time (t)
Demodulated wave
time (t)

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to PWM modulator.
3. Carrier pulse signal is given as another input to PWM modulator.
4. The pulse width modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and pulse width modulated wave.
6. The modulated wave is given as input to demodulator
7. The demodulated output is noted in CRO.
8. The graph is plotted for PWM.

TABULATION:

MODULATION

Signal	Amplitude	Time

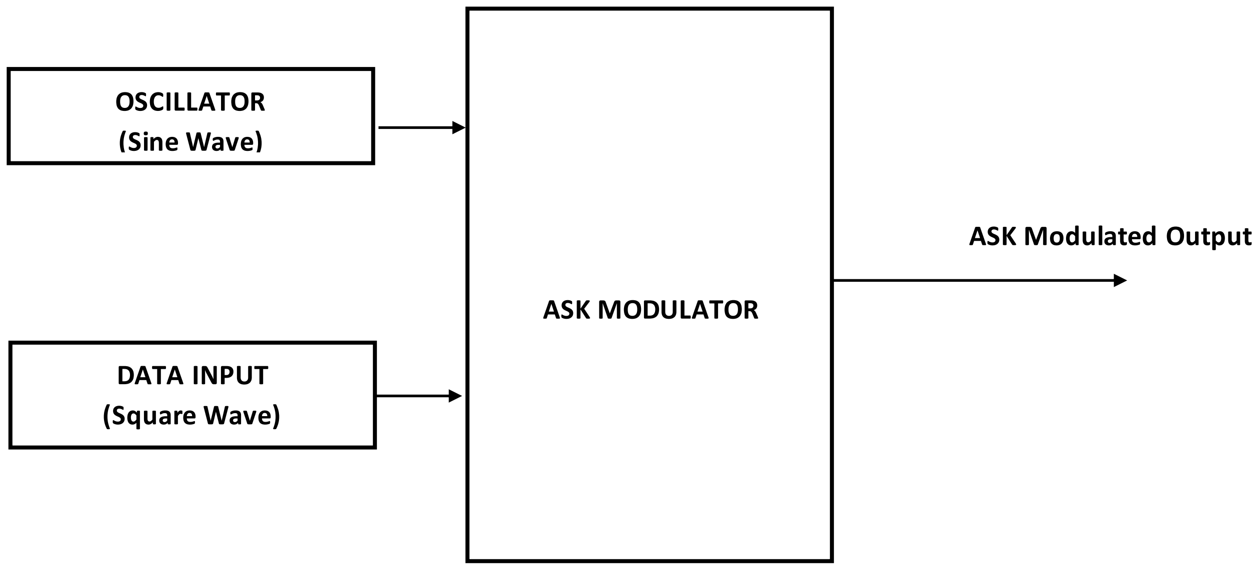
DEMODULATION

Signal	Amplitude	Time

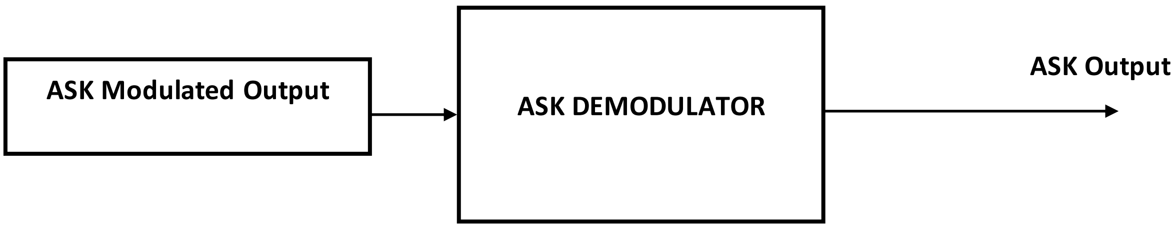
RESULT:

BLOCK DIAGRAM:

ASK MODULATION



ASK DEMODULATION



EXP NO:	AMPLITUDE SHIFT KEYING
DATE	

AIM:

To construct and generate an Amplitude Shift Keying signal and detect the message signal.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	ASK Modulation & Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Amplitude Shift Keying (ASK) is one of the fundamental digital modulation techniques used in communication systems. It involves the modulation of a carrier signal's amplitude based on the digital data being transmitted. The amplitude of the carrier varies while the frequency and phase remain constant. This technique is widely used in various applications such as optical fiber communication, radio frequency (RF) transmission, and low-data-rate wireless communication.

Basic Principle of ASK

ASK is a form of amplitude modulation (AM) where the amplitude of a high-frequency carrier wave is changed in response to digital data. The simplest form of ASK is Binary Amplitude Shift Keying (BASK), where the carrier signal is either present or absent, corresponding to binary values ‘1’ and ‘0’ respectively. Mathematically, the ASK signal can be represented as:

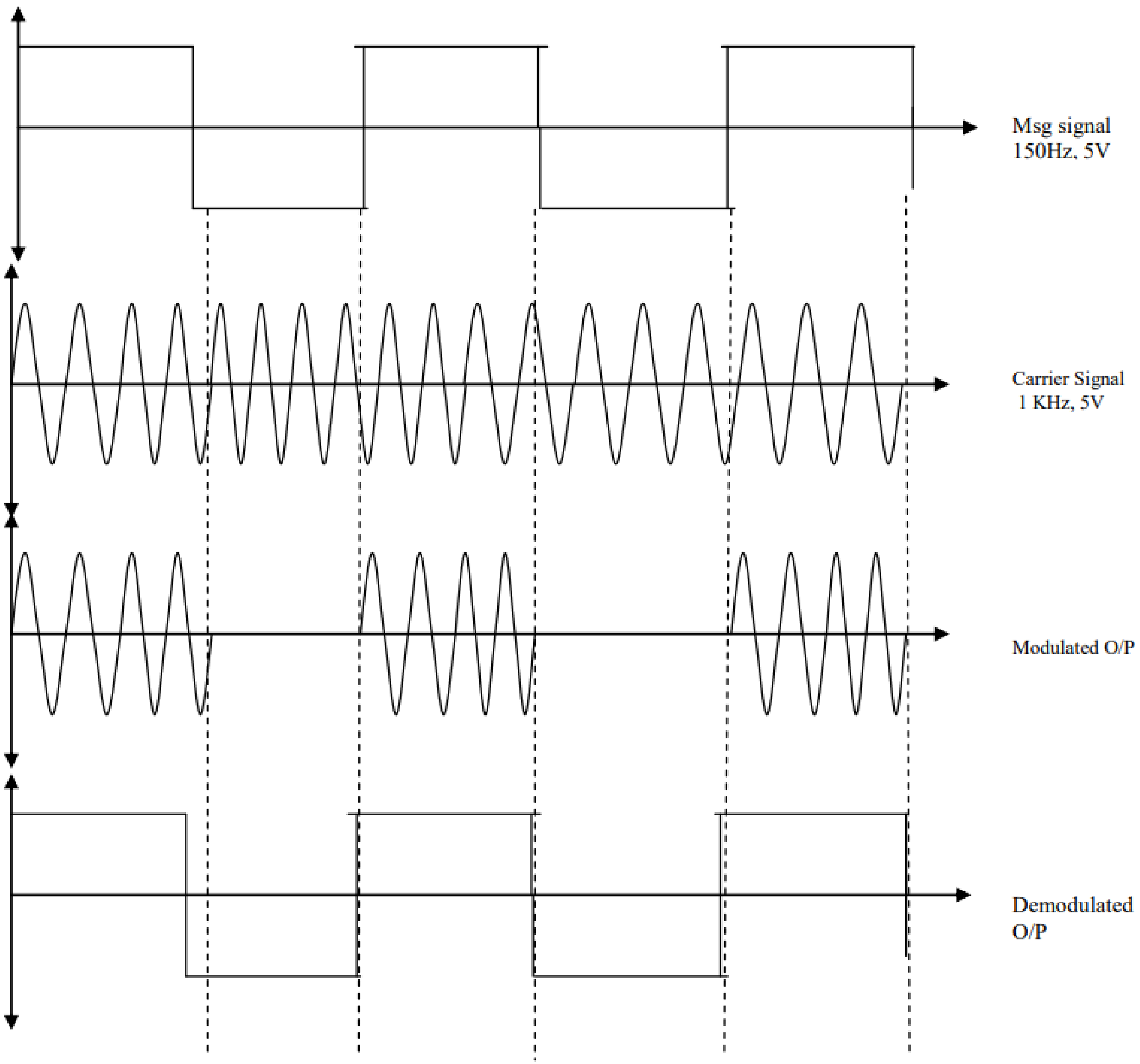
$S(t)= A_c Cos(2\pi ft)$ for bit ‘1’

$S(t) = 0$ for bit ‘0’

where:

- Ac is the carrier amplitude (constant),
- f is the instantaneous frequency that changes according to the input data,
- t represents time.

MODEL GRAPH:


Msg signal
150Hz, 5V
Carrier Signal
1 KHz, 5V
Modulated O/P
Demodulated
O/P

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to ASK modulator.
3. Carrier pulse signal is given as another input to ASK modulator.
4. The ASK modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and pulse width modulated wave.
6. The modulated wave is given as input to the demodulator
7. The demodulated output is noted in the CRO.
8. The graph is plotted for ASK.

TABULATION:

MODULATION

Signal	Amplitude	Time

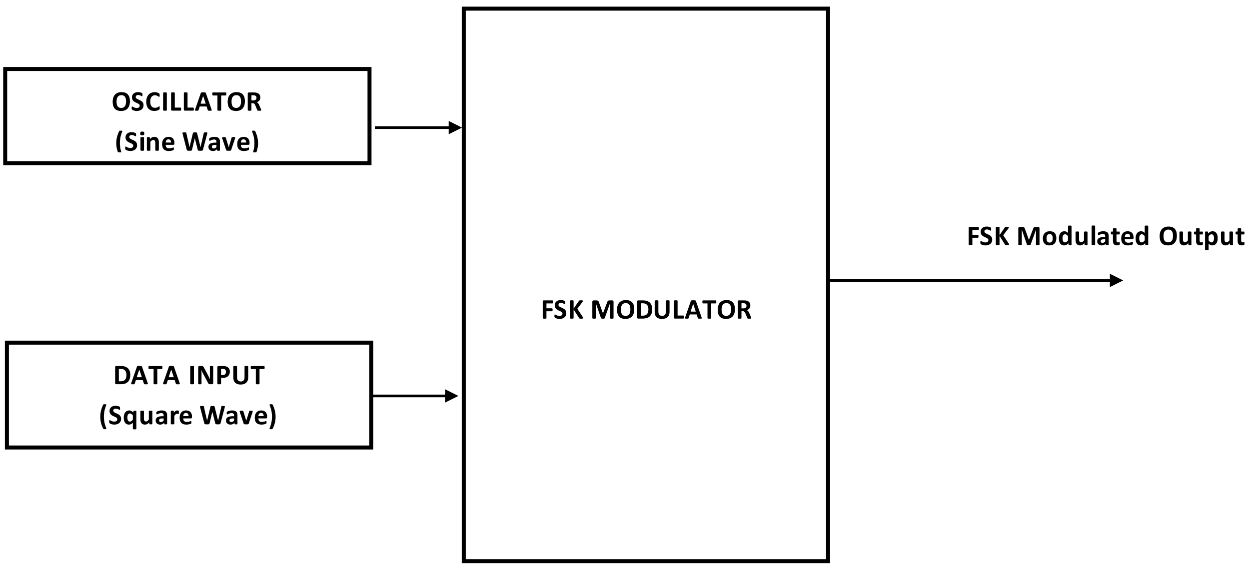
DEMODULATION

Signal	Amplitude	Time

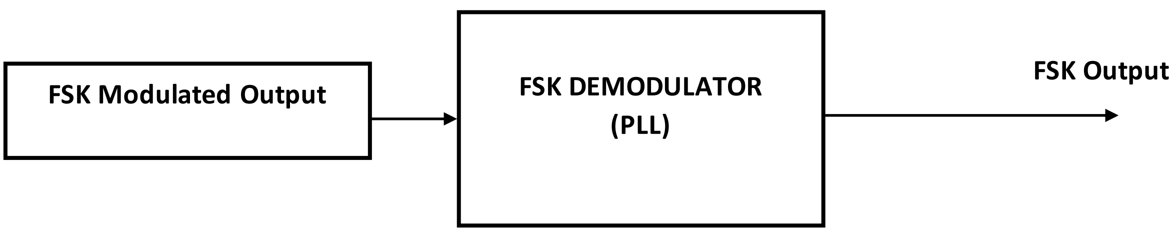
RESULT:

BLOCK DIAGRAM:

FSK MODULATION



FSK DEMODULATION



EXP NO:	FREQUENCY SHIFT KEYING
DATE	

AIM:

To construct and generate a Frequency Shift Keying signal and detect the message signal.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	FSK Modulation & Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Frequency Shift Keying (FSK) is one of the most fundamental digital modulation techniques used in communication systems. It involves shifting the frequency of a carrier wave to transmit digital data. Unlike Amplitude Shift Keying (ASK), which varies the amplitude, or Phase Shift Keying (PSK), which varies the phase, FSK modifies the frequency of the carrier to represent binary data. FSK is widely used in applications such as radio transmission, telemetry, fax machines, RFID systems, and Bluetooth technology. Its resistance to noise and ability to perform well in various communication environments make it a popular choice in digital communication.

Basic Principle of FSK

FSK encodes data by shifting the frequency of a carrier signal between two or more distinct values. The mathematical representation of an FSK-modulated signal is:

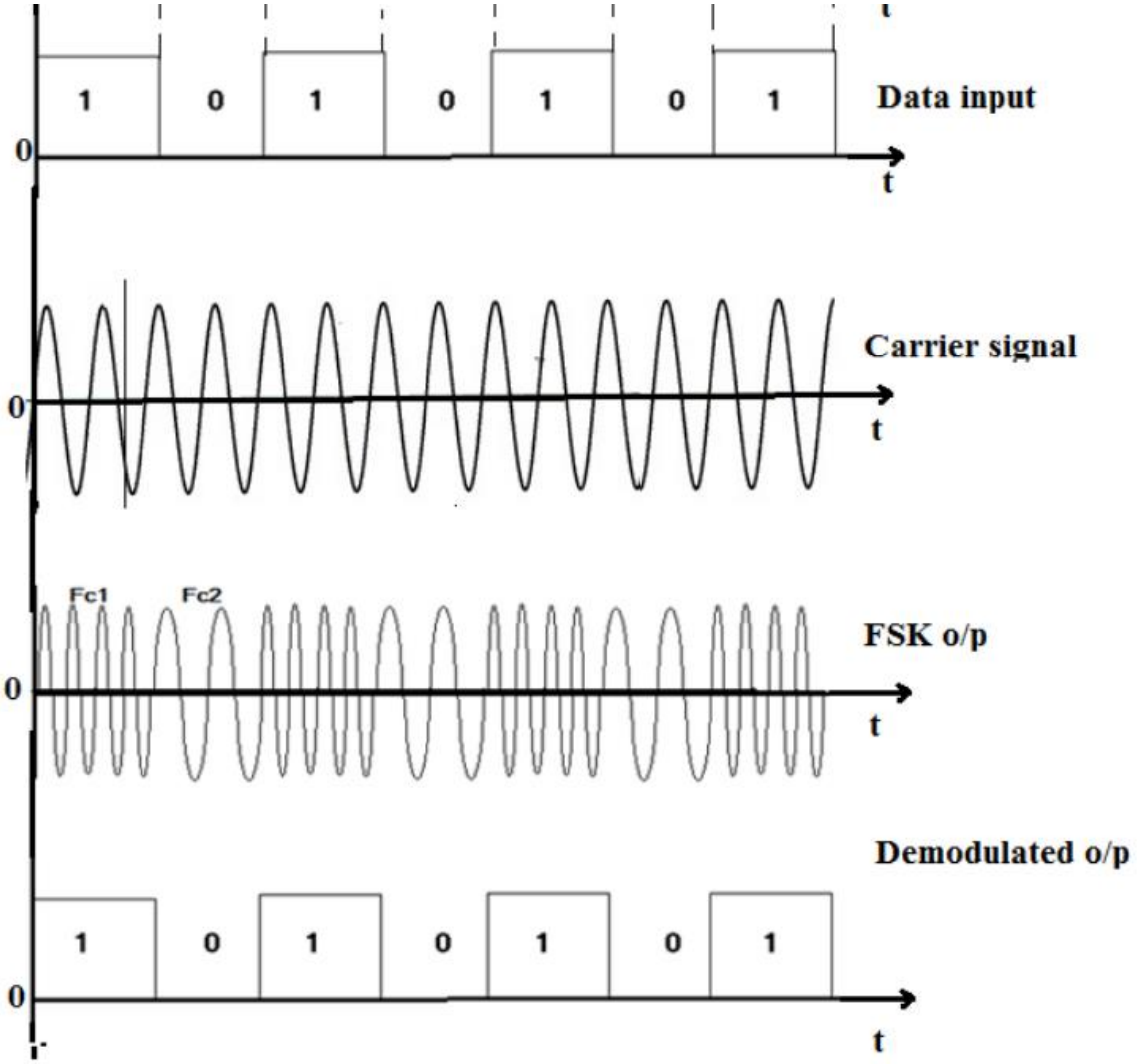
FSK encodes data by shifting the frequency of a carrier signal between two or more distinct values. The mathematical representation of an FSK-modulated signal is:

$$S(t) = A_c \cos(2\pi f t)$$

where:

- Ac is the carrier amplitude (constant),
- f is the instantaneous frequency that changes according to the input data,
- t represents time.

MODEL GRAPH:


1
0
1
0
1
0
1
Data input
0
t
Carrier signal
0
t
Fc1
Fc2
FSK o/p
0
t
Demodulated o/p
1
0
1
0
1
0
1
0
t

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to FSK modulator.
3. Carrier pulse signal is given as another input to FSK modulator.
4. The ASK modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and pulse width modulated wave.
6. The modulated wave is given as input to demodulator
7. The demodulated output is noted in CRO.
8. The graph is plotted for FSK.

TABULATION:

MODULATION

Signal	Amplitude	Time

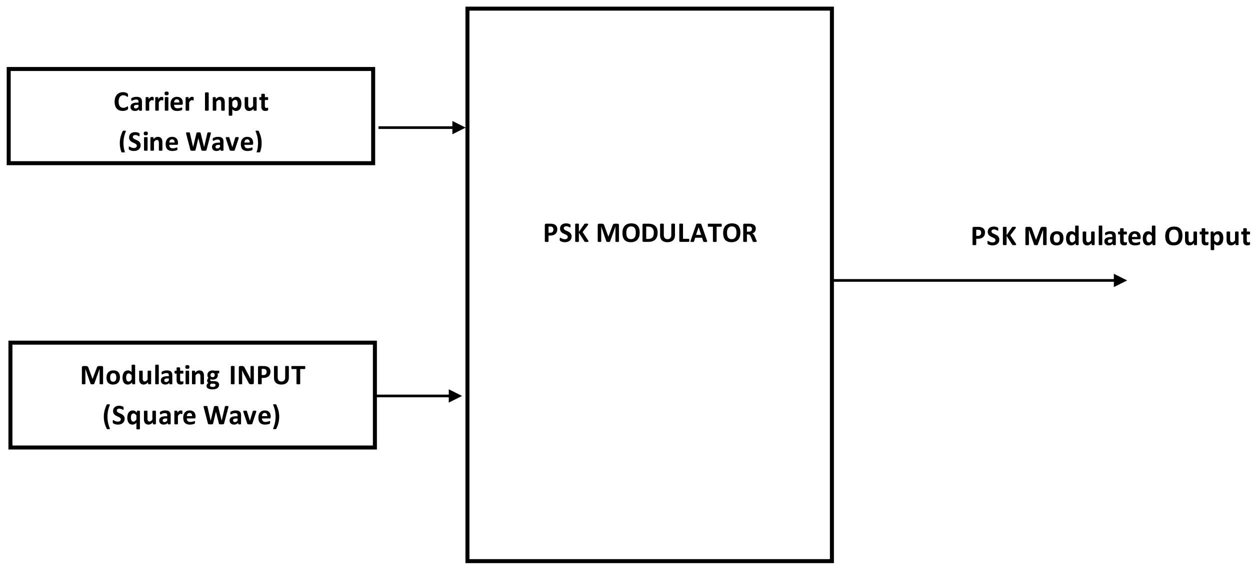
DEMODULATION

Signal	Amplitude	Time

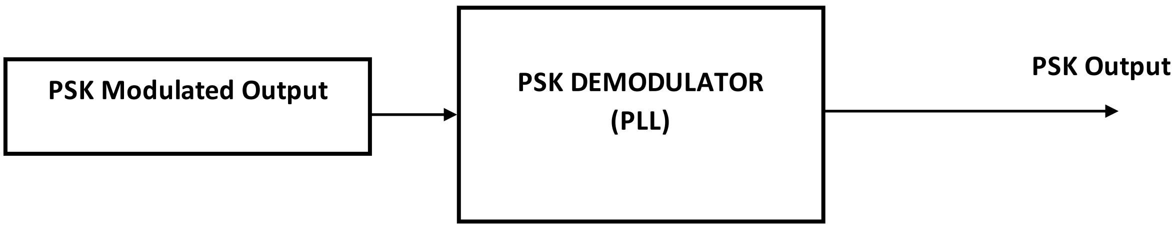
RESULT:

BLOCK DIAGRAM:

PSK MODULATION



PSK DEMODULATION



EXP NO:	PHASE SHIFT KEYING
DATE	

AIM:

To construct and generate a Phase Shift Keying signal and detect the message signal.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	PSK Modulation & Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Phase Shift Keying (PSK) is one of the most widely used digital modulation techniques in modern communication systems. It involves varying the phase of a carrier signal to represent digital data. Unlike Amplitude Shift Keying (ASK), which modifies the signal's amplitude, PSK maintains a constant amplitude and changes only the phase. This makes PSK more resistant to noise and more efficient for high-speed data transmission.

Basic Principle of PSK

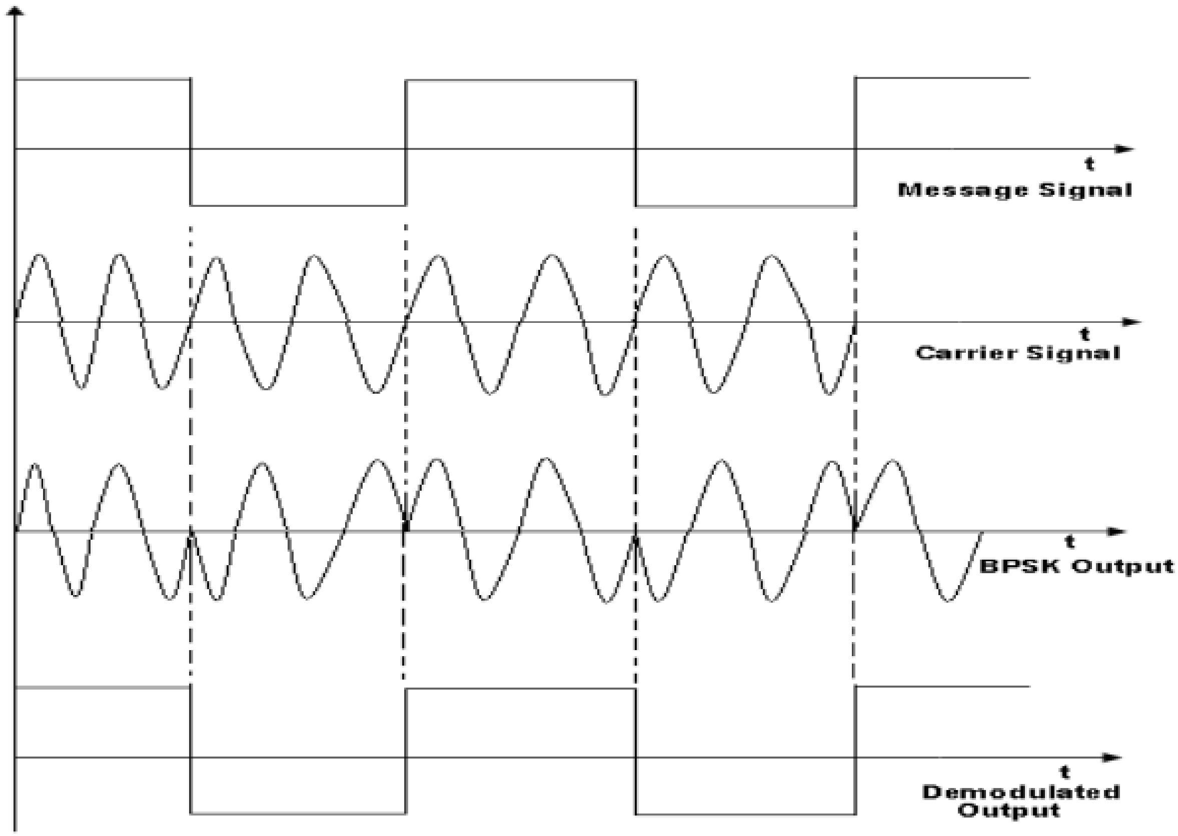
In PSK, the phase of the carrier wave is modified according to the digital data being transmitted. The general mathematical representation of a PSK signal is:

$$\mathbf{S(t)= A_c Cos(2\pi ft + Ø)}$$

where:

- Ac is the carrier amplitude (constant),
- fc is the carrier frequency,
- **Ø** is the phase shift applied based on the input data,
- t represents time.

MODEL GRAPH:


t
Message Signal
t
Carrier Signal
t
BPSK Output
t
Demodulated Output

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to PSK modulator.
3. Carrier pulse signal is given as another input to PSK modulator.
4. The ASK modulated waveform obtained is viewed in CRO.
5. Readings are taken for message, carrier and pulse width modulated wave.
6. The modulated wave is given as input to demodulator
7. The demodulated output is noted in CRO.
8. The graph is plotted for PSK.

TABULATION:

MODULATION

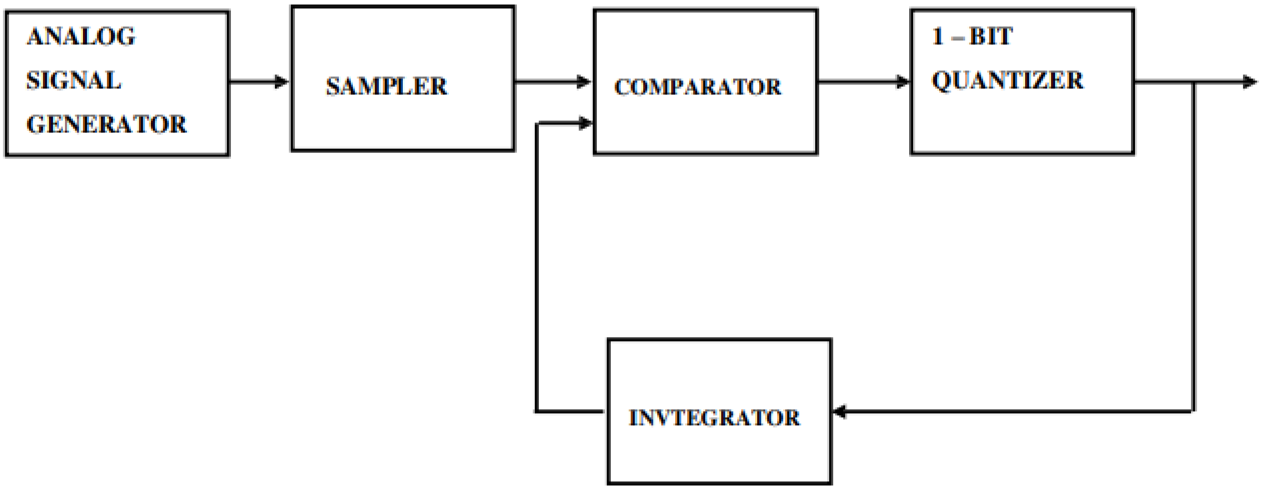
Signal	Amplitude	Time

DEMODULATION

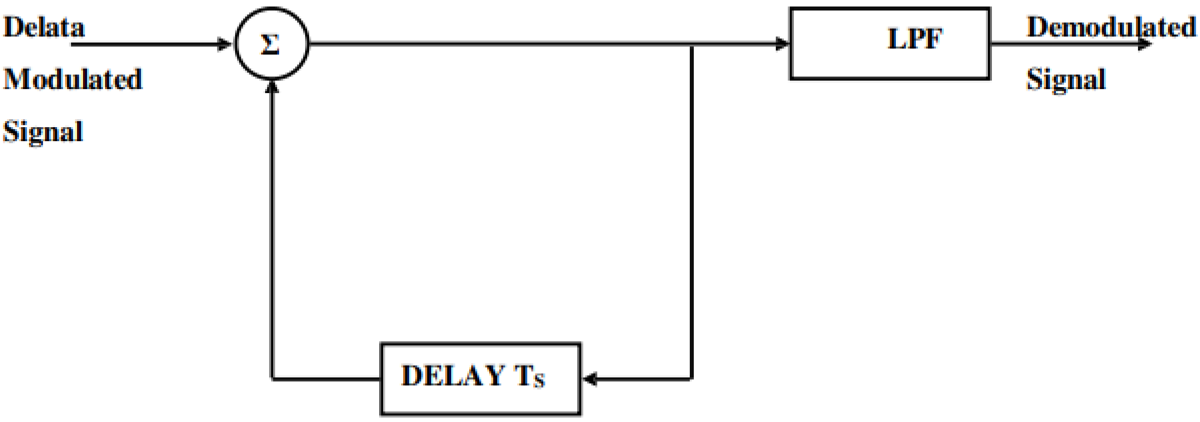
Signal	Amplitude	Time

RESULT:

DM – MODULATOR



DM – DEMODULATOR



EXP NO:	DELTA MODULATION AND DEMODULATION
DATE	

AIM:

To obtain Delta Modulated and Demodulated signal for give message signal.

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	Delta Modulation & Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Delta modulation

Delta Modulation (DM) is a type of pulse-code modulation (PCM) that encodes an analog signal into a digital format using a one-bit data stream. Instead of encoding the absolute amplitude of the signal, DM encodes the difference between consecutive samples, making it a simpler and more bandwidth-efficient method of digital transmission.

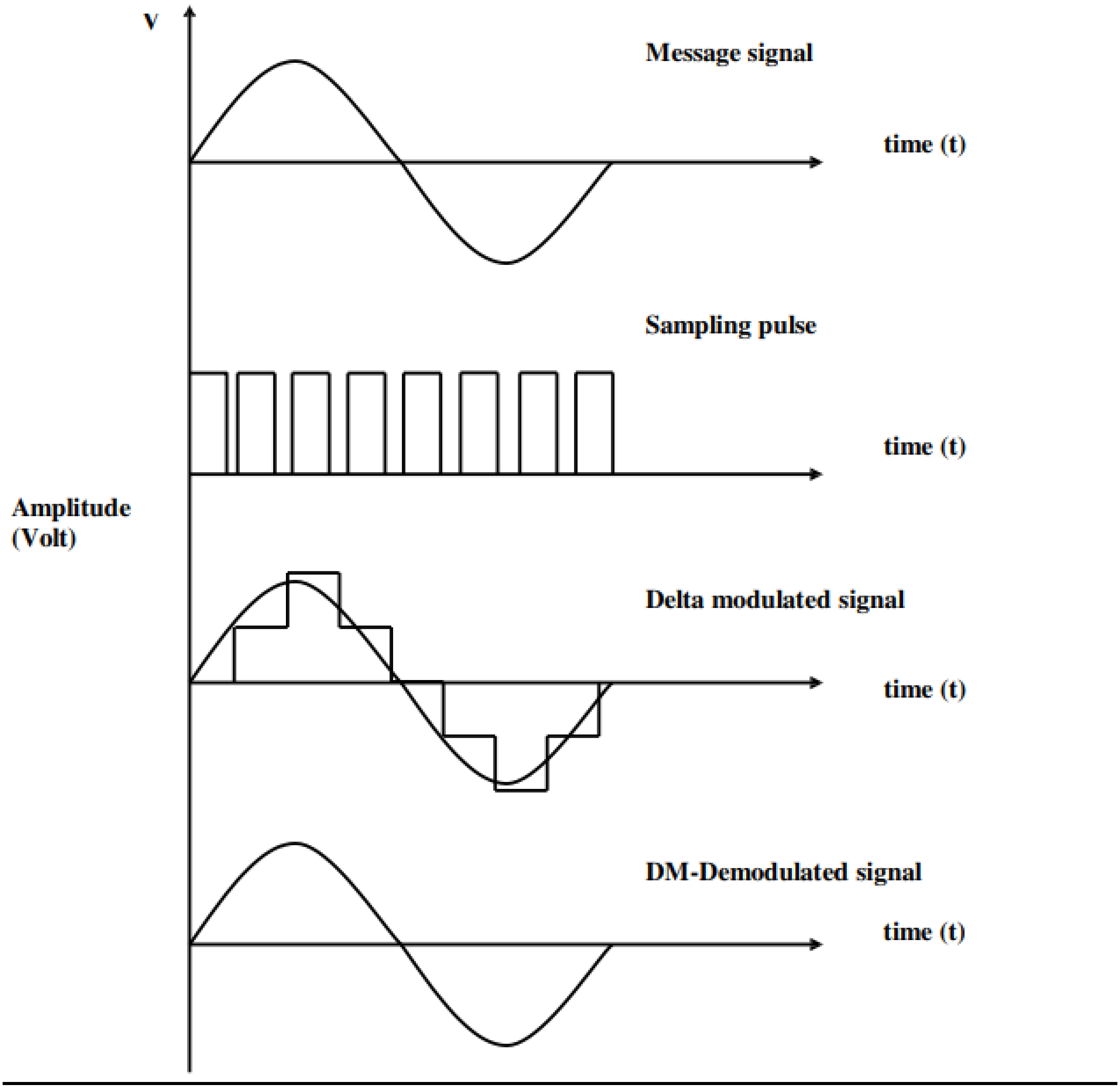
Working Principle of Delta Modulation

1. The input analog signal is sampled at a high rate.
2. The difference between the current and previous sample is calculated.
3. A comparator determines if the signal is increasing or decreasing.
4. If the signal increases, a binary '1' is transmitted; if it decreases, a '0' is transmitted.
5. A 1-bit quantizer and an integrator reconstruct the signal at the receiver.

Delta Demodulation

- At the receiver, a demodulator integrates the binary data to reconstruct the original waveform.
- A low-pass filter smooths out the reconstructed signal to match the original input as closely as possible.

MODEL GRAPH:


V
Message signal
time (t)
Sampling pulse
time (t)
Amplitude
(Volt)
Delta modulated signal
time (t)
DM-Demodulated signal
time (t)

PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to DM modulator circuit
3. pulse signal is given as another input to DM modulator circuit
4. The delta modulated waveform obtained is viewed in CRO
5. Readings are taken for message, pulse and DM wave.
6. The delta modulated wave is given as input to demodulator circuit.
7. The demodulated output is noted in CRO.
8. The graph is plotted for delta modulated and demodulated wave.

TABULATION:

MODULATION

Signal	Amplitude	Time

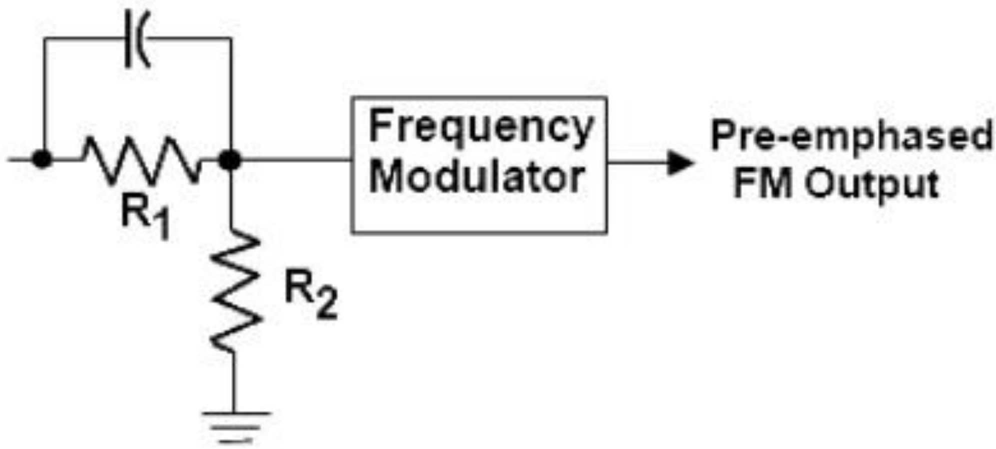
DEMODULATION

Signal	Amplitude	Time

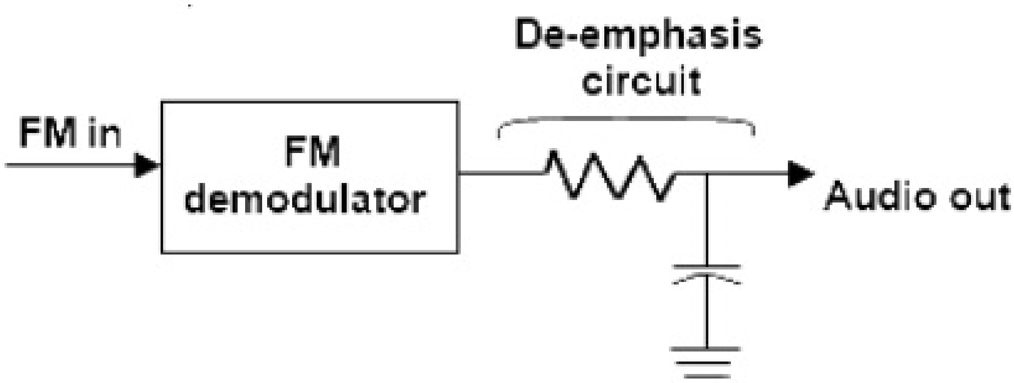
RESULT:

CIRCUIT DIAGRAM:

PRE- EMPHASIS:



DE- EMPHASIS:



EXP NO:	PRE –EMPHASIS AND DE-EMPHASIS
DATE	

AIM:

To observe the effects of pre-emphasis & de-emphasis on given input signal

APPARATUS REQUIRED:

S.No	Hardware Requirements	Quantity
1	Delta Modulation & Demodulation Kit	1
2	DSO	1
3	Probe	Few
4	Patch cord	Few

THEORY:

Pre-Emphasis and De-Emphasis are signal processing techniques used in communication systems to improve signal-to-noise ratio (SNR) and reduce distortion, particularly in frequency modulation (FM) and audio transmission.

Pre-Emphasis

- Pre-Emphasis is a technique where higher frequency components of a signal are amplified before transmission.
- This is done to compensate for high-frequency attenuation caused by noise and channel imperfections.
- A high-pass filter is used in the transmitter to boost high frequencies.

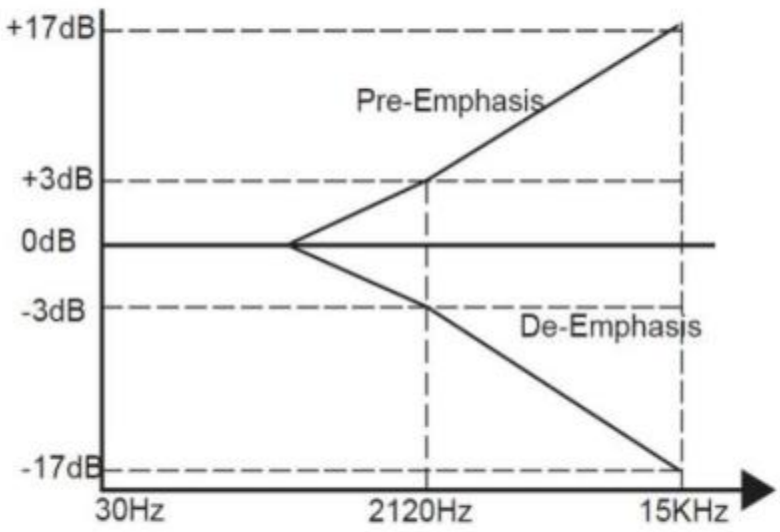
De-Emphasis

- De-Emphasis is the inverse process of Pre-Emphasis and is applied at the receiver.
- A low-pass filter is used to restore the original signal by attenuating the boosted high frequencies.
- This helps in reducing noise interference that primarily affects high frequencies.

Need for Pre-Emphasis and De-Emphasis

- In FM transmission, noise affects higher frequencies more than lower ones.
- Pre-Emphasis boosts high frequencies before transmission, and De-Emphasis attenuates them at the receiver, thereby minimizing noise impact.

MODEL GRAPH:



PROCEDURE:

1. Connections must be given as per the diagram.
2. Low frequency message signal is given as one input to DM modulator circuit
3. pulse signal is given as another input to DM modulator circuit
4. The delta modulated waveform obtained is viewed in CRO
5. Readings are taken for message, pulse and DM wave.
6. The delta modulated wave is given as input to demodulator circuit.
7. The demodulated output is noted in CRO.
8. The graph is plotted for delta modulated and demodulated wave.

TABULATION:

Pre-Emphasis

Frequency (Hz)	I/p voltage (Vi)	O/p voltage (Vo)	Gain = 20log (Vo/Vi)

De-Emphasis

Frequency (Hz)	I/p voltage (Vi)	O/p voltage (Vo)	Gain = 20log (Vo/Vi)

RESULT:

SIMULATION EXPERIMENTS

EXP NO:	Simulation of ASK, FSK, and BPSK Generation and Detection Schemes
DATE	

AIM:

To simulate and analyze the generation and detection schemes of Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Binary Phase Shift Keying (BPSK) using Scilab

APPARATUS REQUIRED:

S.No	Requirements	Quantity
1	Personal Computer	1
2	Scilab 2025 Version	1

THEORY:

Digital modulation techniques are used to transmit digital information over communication channels efficiently. The three fundamental modulation techniques studied in this experiment are:

1. Amplitude Shift Keying (ASK)

ASK is a digital modulation technique where the amplitude of a carrier wave is varied in accordance with the binary data signal. The modulated signal takes two amplitude levels, corresponding to binary '1' and '0'.

- **Modulation:** The carrier wave is multiplied by the binary data.
- **Demodulation:** Envelope detection or coherent detection is used to recover the original binary data.
- **Advantages:** Simple implementation, low power consumption.
- **Disadvantages:** High susceptibility to noise and signal fading.

2. Frequency Shift Keying (FSK)

FSK is a modulation scheme where the frequency of the carrier signal is changed between two distinct values representing binary '1' and '0'.

- **Modulation:** The frequency of the carrier wave is shifted based on input binary data.
- **Demodulation:** Frequency discrimination or phase-locked loop (PLL) techniques are used to detect the original signal.
- **Advantages:** Better noise immunity compared to ASK.
- **Disadvantages:** Requires more bandwidth than ASK.

3. Binary Phase Shift Keying (BPSK)

BPSK is a modulation scheme where the phase of the carrier wave is shifted between two values (0° and 180°) based on the binary data.

- **Modulation:** A carrier wave undergoes a phase shift of 0° or 180° depending on the binary input.
- **Demodulation:** Coherent detection using a matched filter or a phase-locked loop is employed to retrieve the original data.
- **Advantages:** High noise immunity, efficient power usage.
- **Disadvantages:** Requires complex synchronization at the receiver.

PROGRAM:

1. Amplitude Shift Keying (ASK)

```
clc;
clear;
close();

// Parameters
t = 0:0.001:1;   // Time vector for one bit duration
fc = 5;          // Carrier frequency
fs = 1000;       // Sampling frequency
bit_stream = [1 0 1 1 0 1 0 0 1];  // Binary Data
samples_per_bit = length(t);       // Samples per bit

// Carrier Signal
carrier = sin(2 * %pi * fc * t);

// Initialize Signals
ask_signal = [];
input_signal = [];
carrier_wave = [];
demod_signal = [];

// ASK Modulation
for i = 1:length(bit_stream)
   if bit_stream(i) == 1 then
      ask_signal = [ask_signal, carrier];  // Bit '1' -> Carrier wave
      input_signal = [input_signal, ones(1, samples_per_bit)]; // High level
   else
      ask_signal = [ask_signal, zeros(1, samples_per_bit)];  // Bit '0' -> No signal
      input_signal = [input_signal, zeros(1, samples_per_bit)]; // Low level
   end
   carrier_wave = [carrier_wave, carrier]; // Continuous carrier signal
end

// ASK Demodulation (Envelope Detection)
for i = 1:length(bit_stream)
   start_idx = (i-1) * samples_per_bit + 1;
   end_idx = i * samples_per_bit;

   if end_idx > length(ask_signal) then
```

```
        break;
    end

    segment = ask_signal(start_idx:end_idx); // Extract segment
    energy = sum(segment.^2);  // Compute energy

    // Threshold detection
    if energy > 0.5 then
        demod_signal = [demod_signal, ones(1, samples_per_bit)]; // High level
    else
        demod_signal = [demod_signal, zeros(1, samples_per_bit)]; // Low level
    end
end

// Plot Signals in Single Window
subplot(4,1,1);
plot(input_signal, 'k');
title('Modulating Input Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(input_signal), 1.2];  // Set Y-axis limits

subplot(4,1,2);
plot(carrier_wave, 'b');
title('Carrier Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,3);
plot(ask_signal, 'r');
title('ASK Modulated Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,4);
plot(demod_signal, 'g');
title('ASK Demodulated Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(demod_signal), 1.2];  // Set Y-axis limits
```

2. Frequency Shift Keying (FSK)

```
clc;
clear;
close();

// Parameters
t = 0:0.001:1;   // Time vector for one bit duration
f1 = 5;          // Frequency for bit 1
f0 = 2;          // Frequency for bit 0
fs = 1000;       // Sampling frequency
bit_stream = [1 0 1 1 0 1 0 0 1];  // Binary Data
samples_per_bit = length(t);       // Samples per bit

// Carrier Signals
carrier1 = sin(2 * %pi * f1 * t); // High frequency for bit 1
carrier0 = sin(2 * %pi * f0 * t); // Low frequency for bit 0

// Initialize Signals
fsk_signal = [];
input_signal = [];
carrier_wave = [];
demod_signal = [];

// FSK Modulation
for i = 1:length(bit_stream)
   if bit_stream(i) == 1 then
      fsk_signal = [fsk_signal, carrier1];  // Bit '1' -> High frequency
      input_signal = [input_signal, ones(1, samples_per_bit)]; // High level
   else
      fsk_signal = [fsk_signal, carrier0];  // Bit '0' -> Low frequency
      input_signal = [input_signal, zeros(1, samples_per_bit)]; // Low level
   end
end

// FSK Demodulation (Energy Detection)
for i = 1:length(bit_stream)
   start_idx = (i-1) * samples_per_bit + 1;
   end_idx = i * samples_per_bit;

   if end_idx > length(fsk_signal) then
```

```
        break;
    end

    segment = fsk_signal(start_idx:end_idx); // Extract segment
    energy1 = sum((segment .* carrier1).^2);  // Correlation with f1
    energy0 = sum((segment .* carrier0).^2);  // Correlation with f0

    // Decision Making
    if energy1 > energy0 then
        demod_signal = [demod_signal, ones(1, samples_per_bit)]; // Bit '1'
    else
        demod_signal = [demod_signal, zeros(1, samples_per_bit)]; // Bit '0'
    end
end

// Plot Signals in Single Window
subplot(4,1,1);
plot(input_signal, 'k');
title('Modulating Input Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(input_signal), 1.2];

subplot(4,1,2);
plot(fsk_signal, 'b');
title('FSK Modulated Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,3);
plot(fsk_signal, 'r');
title('FSK Received Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,4);
plot(demod_signal, 'g');
title('FSK Demodulated Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(demod_signal), 1.2];
```

3. Binary Phase Shift Keying (BPSK)

```
clc;
clear;
close();

// Parameters
t = 0:0.001:1;   // Time vector for one bit duration
fc = 5;          // Carrier frequency
fs = 1000;       // Sampling frequency
bit_stream = [1 0 1 1 0 1 0 0 1];  // Binary Data
samples_per_bit = length(t);       // Samples per bit

// Carrier Signal
carrier = sin(2 * %pi * fc * t);

// Initialize Signals
bpsk_signal = [];
input_signal = [];
demod_signal = [];

// BPSK Modulation
for i = 1:length(bit_stream)
   if bit_stream(i) == 1 then
      bpsk_signal = [bpsk_signal, carrier];  // Bit '1' -> Normal carrier
      input_signal = [input_signal, ones(1, samples_per_bit)]; // High level
   else
      bpsk_signal = [bpsk_signal, -carrier];  // Bit '0' -> Inverted carrier
      input_signal = [input_signal, zeros(1, samples_per_bit)]; // Low level
   end
end

// BPSK Demodulation (Correlation)
for i = 1:length(bit_stream)
   start_idx = (i-1) * samples_per_bit + 1;
   end_idx = i * samples_per_bit;

   if end_idx > length(bpsk_signal) then
      break;
   end

   segment = bpsk_signal(start_idx:end_idx); // Extract segment
```

```
    correlation = sum(segment .* carrier);  // Correlation with carrier

    // Decision Making
    if correlation > 0 then
        demod_signal = [demod_signal, ones(1, samples_per_bit)]; // Bit '1'
    else
        demod_signal = [demod_signal, zeros(1, samples_per_bit)]; // Bit '0'
    end
end

// Plot Signals in Single Window
subplot(4,1,1);
plot(input_signal, 'k');
title('Modulating Input Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(input_signal), 1.2];

subplot(4,1,2);
plot(carrier, 'b');
title('Carrier Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,3);
plot(bpsk_signal, 'r');
title('BPSK Modulated Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,4);
plot(demod_signal, 'g');
title('BPSK Demodulated Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(demod_signal), 1.2];
```

PROCEDURE

1. **Open Scilab:** Launch Scilab on your system.
2. **Create a New Script:** Open the SciNotes editor by clicking on Applications > SciNotes.
3. **Write the Code:** Enter the Scilab script for generating ASK, FSK, and BPSK signals.
4. **Save the File:** Save the script with a .sci extension, e.g., modulation.sci.
5. **Load the Script in Scilab:** In the Scilab console, navigate to the script's directory and type: exec('modulation.sci', -1);
6. **Run the Simulation:** Execute the script, and Scilab will generate modulation and demodulation plots.
7. **Analyze the Results:** Observe the ASK, FSK, and BPSK waveforms in the generated figures.

RESULT:

EXP NO:	**Simulation of DPSK, QPSK and QAM Generation and Detection Schemes.**
DATE	

AIM:
To simulate and analyze the generation and detection schemes of Differential Phase Shift Keying (DPSK), Quadrature Phase Shift Keying (QPSK), and Quadrature Amplitude Modulation (QAM) using Scilab

APPARATUS REQUIRED:

S.No	Requirements	Quantity
1	Personal Computer	1
2	Scilab 2025 Version	1

THEORY:

Digital modulation techniques are widely used in modern communication systems to transmit information efficiently. The three modulation techniques studied in this experiment are:

1. **Differential Phase Shift Keying (DPSK)**

DPSK is a variant of PSK where the phase of the carrier signal is changed relative to the previous signal rather than a fixed reference.

- Modulation: The phase of the carrier signal is changed based on the difference between consecutive bits.
- Demodulation: Uses a differential decoder to compare the phase of consecutive received symbols.
- Advantages: Eliminates the need for a coherent receiver.
- Disadvantages: More susceptible to noise compared to coherent PSK.

2. Quadrature Phase Shift Keying (QPSK)

QPSK is a phase modulation scheme that encodes two bits per symbol, making it more bandwidth-efficient than BPSK.

- Modulation: Uses four different phase shifts (0°, 90°, 180°, and 270°) to represent bit pairs.
- Demodulation: A coherent receiver detects the phase shifts and maps them back to the original bit sequence.
- Advantages: Higher data rate compared to BPSK.
- Disadvantages: More complex receiver design.

3. Quadrature Amplitude Modulation (QAM)

QAM is a hybrid modulation technique that combines both amplitude and phase variations to represent multiple bits per symbol.

- Modulation: Uses a combination of amplitude levels and phase shifts to encode data.
- Demodulation: Requires a coherent receiver to decode both amplitude and phase information.
- Advantages: High spectral efficiency, widely used in broadband communication.
- Disadvantages: More susceptible to noise and requires complex equalization.

PROGRAM

1. Differential Phase Shift Keying (DPSK)

```
clc;
clear;
close();

// Parameters
t = 0:0.001:1;   // Time vector for one bit duration
fc = 5;          // Carrier frequency
fs = 1000;       // Sampling frequency
bit_stream = [1 0 1 1 0 1 0 0 1];  // Binary Data
samples_per_bit = length(t);       // Samples per bit

// Carrier Signal
carrier = sin(2 * %pi * fc * t);

// Initialize Signals
dpsk_signal = [];
input_signal = [];
demod_signal = [];

// DPSK Modulation (Differential Encoding)
previous_phase = 1;  // Assume initial phase corresponds to bit '1'

for i = 1:length(bit_stream)
   if bit_stream(i) == 1 then
      phase = previous_phase;  // No phase shift if bit is '1'
   else
      phase = -previous_phase;  // 180-degree phase shift if bit is '0'
   end

   modulated_wave = phase * carrier;  // Apply phase shift to carrier
   dpsk_signal = [dpsk_signal, modulated_wave];
   input_signal = [input_signal, bit_stream(i) * ones(1, samples_per_bit)];

   previous_phase = phase;  // Store phase for next bit
end

// DPSK Demodulation (Differential Detection)
previous_bit = 1;  // Assume initial bit was '1'
```

```
for i = 1:length(bit_stream)
    start_idx = (i-1) * samples_per_bit + 1;
    end_idx = i * samples_per_bit;

    if end_idx > length(dpsk_signal) then
        break;
    end

    segment = dpsk_signal(start_idx:end_idx); // Extract segment
    correlation = sum(segment .* carrier);  // Correlation with carrier

    // Differential Detection
    if correlation > 0 then
        demod_bit = previous_bit;  // If phase is unchanged, bit remains same
    else
        demod_bit = 1 - previous_bit;  // If phase shift, bit flips
    end

    demod_signal = [demod_signal, demod_bit * ones(1, samples_per_bit)];
    previous_bit = demod_bit;  // Store for next bit
end

// Plot Signals in Single Window
subplot(4,1,1);
plot(input_signal, 'k');
title('Modulating Input Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(input_signal), 1.2];

subplot(4,1,2);
plot(carrier, 'b');
title('Carrier Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,3);
plot(dpsk_signal, 'r');
title('DPSK Modulated Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,4);
plot(demod_signal, 'g');
title('DPSK Demodulated Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(demod_signal), 1.2];
```

2. Quadrature Phase Shift Keying (QPSK)

```
clc;
clear;
close();

// Parameters
t = 0:0.001:1;   // Time vector for one bit duration
fc = 5;          // Carrier frequency
fs = 1000;       // Sampling frequency
bit_stream = [1 0 1 1 0 1 0 0 1 1 0 0];  // Binary Data (even length)
samples_per_bit = length(t);             // Samples per bit

// Ensure binary data length is even (since QPSK uses bit pairs)
if modulo(length(bit_stream), 2) <> 0 then
   bit_stream($+1) = 0;  // Append a zero if needed
end

// Carrier Signals
I_carrier = cos(2 * %pi * fc * t); // In-phase carrier (cosine)
Q_carrier = sin(2 * %pi * fc * t); // Quadrature carrier (sine)

// Initialize Signals
qpsk_signal = [];
input_signal = [];
demod_signal = [];

// QPSK Modulation
for i = 1:2:length(bit_stream)
   b1 = bit_stream(i);
   b2 = bit_stream(i+1);

   // Determine phase based on bit pair
   if b1 == 0 & b2 == 0 then
      phase_shift = %pi/4;  // 45 degrees
   elseif b1 == 0 & b2 == 1 then
      phase_shift = 3*%pi/4;  // 135 degrees
   elseif b1 == 1 & b2 == 1 then
      phase_shift = 5*%pi/4;  // 225 degrees
   else
      phase_shift = 7*%pi/4;  // 315 degrees
   end
```

```
    modulated_wave = cos(2 * %pi * fc * t + phase_shift); // Apply phase shift
    qpsk_signal = [qpsk_signal, modulated_wave];

    input_signal = [input_signal, b1 * ones(1, samples_per_bit)]; // High level for bit 1
    input_signal = [input_signal, b2 * ones(1, samples_per_bit)]; // High level for bit 2
end

// QPSK Demodulation (Coherent Detection)
for i = 1:samples_per_bit:length(qpsk_signal)
    if i + samples_per_bit - 1 > length(qpsk_signal) then
        break;
    end

    segment = qpsk_signal(i:i+samples_per_bit-1);

    // Correlate with I and Q carriers
    I_correlation = sum(segment .* I_carrier);
    Q_correlation = sum(segment .* Q_carrier);

    // Decision Making
    if I_correlation > 0 & Q_correlation > 0 then
        demod_bits = [0 0];  // 45 degrees
    elseif I_correlation < 0 & Q_correlation > 0 then
        demod_bits = [0 1];  // 135 degrees
    elseif I_correlation < 0 & Q_correlation < 0 then
        demod_bits = [1 1];  // 225 degrees
    else
        demod_bits = [1 0];  // 315 degrees
    end

    // Corrected line: Append both bits properly
    demod_signal = [demod_signal, demod_bits(1) * ones(1, samples_per_bit), demod_bits(2) *
ones(1, samples_per_bit)];
end


// Plot Signals in Single Window
subplot(4,1,1);
plot(input_signal, 'k');
title('Modulating Input Signal (Binary Data)');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(input_signal), 1.2];

subplot(4,1,2);
```

```
plot(qpsk_signal, 'b');
title('QPSK Modulated Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,3);
plot(qpsk_signal, 'r');
title('QPSK Received Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,4);
plot(demod_signal, 'g');
title('QPSK Demodulated Signal');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(demod_signal), 1.2];
```

3. Quadrature Amplitude Modulation:

```
clc;
clear;
close();

// Parameters
t = 0:0.001:1;   // Time vector for one bit duration
fc = 5;          // Carrier frequency
fs = 1000;       // Sampling frequency
bit_stream = [1 0 1  0 1 0  1 1 0  0 0 1  1 1 1  0 0 0];  // Binary Data (multiple of 3)
samples_per_symbol = length(t);  // Samples per symbol (3 bits per QAM symbol)

// Ensure binary data length is a multiple of 3
if modulo(length(bit_stream), 3) <> 0 then
   bit_stream($+1:$+3-modulo(length(bit_stream),3)) = 0;  // Append zeros if needed
end

//  Define `input_signal` for plotting
input_signal = bit_stream;

// 8-QAM Constellation Mapping (Gray Encoding)
qam_map = [ -3 -1; -1 -3; 1 -3; 3 -1;
           -1  3; -3  1; 1  3; 3  1];

//  Initialize modulated signal
qam_signal = [];

for i = 1:3:length(bit_stream)
   b1 = bit_stream(i);
   b2 = bit_stream(i+1);
   b3 = bit_stream(i+2);

   symbol_index = b1*4 + b2*2 + b3 + 1;  // Convert bits to index
   I = qam_map(symbol_index, 1);  // In-phase
   Q = qam_map(symbol_index, 2);  // Quadrature

   modulated_wave = I * cos(2 * %pi * fc * t) + Q * sin(2 * %pi * fc * t);
   qam_signal = [qam_signal, modulated_wave];  // Store modulated signal
end

// Initialize `demod_signal` before use
demod_signal = [];
```

```
// 8-QAM Demodulation
for i = 1:samples_per_symbol:length(qam_signal)
   if i + samples_per_symbol - 1 > length(qam_signal) then
      break;
   end

   segment = qam_signal(i:i+samples_per_symbol-1);

   // Ensure correct correlation calculations
   I_correlation = sum(segment .* cos(2 * %pi * fc * t));
   Q_correlation = sum(segment .* sin(2 * %pi * fc * t));

   // Find the closest constellation point
   distances = (qam_map(:,1) - I_correlation).^2 + (qam_map(:,2) - Q_correlation).^2;
   [closest_value, closest_index] = min(distances); // Ensure `closest_index` exists

   // Convert index to 3-bit binary & fix data format
   demod_bits = ascii(dec2bin(closest_index-1, 3)) - 48;
demod_signal = [];  // Initialize before loop
   demod_signal = [demod_signal, demod_bits(1) * ones(1, samples_per_symbol),
                        demod_bits(2) * ones(1, samples_per_symbol),
                        demod_bits(3) * ones(1, samples_per_symbol)];
end

//  Plot Signals in a Single Window
subplot(4,1,1);
plot(input_signal, 'k');
title('Modulating Input Signal (Binary Data)');
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(input_signal), 1.2];

subplot(4,1,2);
plot(qam_signal, 'b');
title('8-QAM Modulated Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,3);
plot(qam_signal, 'r');
title('8-QAM Received Signal');
xlabel('Time'); ylabel('Amplitude');

subplot(4,1,4);
plot(demod_signal, 'g');
title('8-QAM Demodulated Signal');
```

```
xlabel('Time'); ylabel('Amplitude');
gca().data_bounds = [0, -0.2; length(demod_signal), 1.2];
```

PROCEDURE:

1. Open Scilab: Launch Scilab on your system.
2. Create a New Script: Open SciNotes by selecting Applications > SciNotes.
3. Write the Code: Enter the Scilab script for generating and detecting DPSK, QPSK, and QAM signals.
4. Save the File: Save the script with a .sci extension, e.g., modulation_dpsk_qpsk_qam.sci.
5. Load the Script in Scilab: In the Scilab console, navigate to the script’s directory and type: exec('modulation_dpsk_qpsk_qam.sci', -1);
6. Run the Simulation: Execute the script, and Scilab will generate the modulation and demodulation plots.
7. Analyze the Results: Observe the waveforms and compare the performance of DPSK, QPSK, and QAM in terms of noise immunity and bandwidth efficiency.

RESULT:

EXP NO:	Simulation of Linear Block and Cyclic Error Control coding Schemes
DATE	

AIM:

To simulate and analyze the encoding and decoding processes of Linear Block Codes and Cyclic Codes for error detection and correction using Scilab.

APPARATUS REQUIRED:

S.No	Requirements	Quantity
1	Personal Computer	1
2	Scilab 2025 Version	1

THEORY:

Error control coding is essential in digital communication systems to detect and correct errors introduced during transmission. The two primary coding schemes analyzed in this experiment are:

1. Linear Block Codes

Linear Block Codes are a class of error-correcting codes where each codeword is a linear combination of message bits.

- **Encoding:** A message of length is multiplied by a generator matrix to generate a codeword of length (where).
- **Decoding:** The received codeword is checked using a parity-check matrix to detect and correct errors.
- **Advantages:** Simple encoding and decoding, effective error detection and correction.
- **Disadvantages:** Limited error correction capability compared to more advanced coding schemes.

2. Cyclic Codes

Cyclic Codes are a subset of Linear Block Codes where a cyclic shift of a codeword results in another valid codeword.

- **Encoding:** The message polynomial is divided by a generator polynomial, and the remainder is appended to form the codeword.
- **Decoding:** Syndrome decoding or cyclic redundancy check (CRC) is used to detect and correct errors.
- **Advantages:** Efficient encoding and decoding using shift registers.
- **Disadvantages:** More complex than simple parity-based error detection.

PROGRAM:

1. Linear Block coding:

```
clc;
clear;
close();

// Define Parameters
n = 7;  // Codeword length
k = 4;  // Message length

// Generator Matrix for (7,4) Hamming Code (Systematic Form)
G = [1 0 0 0 1 1 0;
    0 1 0 0 1 0 1;
    0 0 1 0 0 1 1;
    0 0 0 1 1 1 1];

// Parity-Check Matrix H (For Error Detection)
H = [1 1 0 1 1 0 0;
    1 0 1 1 0 1 0;
    0 1 1 1 0 0 1];

// Standard Codewords
message = [1 0 1 1;
          0 1 0 1;
          1 1 1 0;
          0 0 1 1];

disp("Original Message Words:");
disp(int32(message));

// Encoding: Codeword = Message * G
codewords = modulo(message * G, 2);
disp("Encoded Codewords:");
disp(int32(codewords));

// Simulate Transmission (Introduce an Error in 1st Codeword)
codewords(1,3) = 1 - codewords(1,3);  // Flipping one bit to introduce an error
disp("Received Codewords (with possible error):");
disp(int32(codewords));

// Compute Syndrome: S = H * C^T
```

```
syndrome = modulo(H * codewords', 2);
disp("Syndrome:");
disp(int32(syndrome'));

// Error Correction
for i = 1:size(codewords, 1)
   syn = syndrome(:, i)';  // Get syndrome for row i
   if norm(syn, 1) ~= 0  // If syndrome is nonzero, an error exists
      for j = 1:n
         if syn == H(:, j)'  // Find the error bit position
            codewords(i, j) = 1 - codewords(i, j);  // Flip the bit
            disp("Error corrected in row " + string(i) + ", bit " + string(j));
            break;
         end
      end
   end
end

disp("Corrected Codewords:");
disp(int32(codewords));

// **Decoding: Extract the First 4 Bits**
decoded_message = codewords(:, 1:k);
disp("Decoded Message:");
disp(int32(decoded_message));

disp("Linear Block Coding with Proper Decoding Completed.");
```

2. Cyclic Error Control coding

```
clc;
clear;
close();

// Generator Polynomial: g(x) = x^3 + x + 1 (1101 in binary)
generator_poly = [1 1 0 1];

// Codeword Length & Message Length
n = 7; // Codeword Length
k = 4; // Message Length

// Sample Message Bits (4-bit message)
message = [1 0 1 1;
        0 1 0 1;
        1 1 1 0;
        0 0 1 1];

disp("Original Message Words:");
disp(int32(message));

// Encoding using manual polynomial division
function codeword=cyclic_encode(msg, gen_poly, n)
   k = length(msg);
   padded_msg = [msg, zeros(1, n-k)];  // Append n-k (3) zeros

   for i = 1:k
      if padded_msg(i) == 1 then
         padded_msg(i:i+length(gen_poly)-1) = modulo(padded_msg(i:i+length(gen_poly)-1) - gen_poly, 2);
      end
   end

   codeword = [msg, padded_msg(k+1:$)];  // Append remainder
endfunction

encoded_codewords = [];
for i = 1:size(message,1)
   encoded_codewords = [encoded_codewords; cyclic_encode(message(i,:), generator_poly, n)];
end

disp("Encoded Codewords:");
```

```
disp(int32(encoded_codewords));

// Extract Original Message (First 4 Bits)
decoded_messages = encoded_codewords(:, 1:k);
disp("Decoded Messages:");
disp(int32(decoded_messages));
```

PROCEDURE:

1. **Open Scilab:** Launch Scilab on your system.
2. **Create a New Script:** Open SciNotes by selecting Applications > SciNotes.
3. **Write the Code:** Enter the Scilab script for implementing Linear Block and Cyclic Codes for encoding and decoding.
4. **Save the File:** Save the script with a .sci extension, e.g., error_control_coding.sci.
5. **Load the Script in Scilab:** In the Scilab console, navigate to the script’s directory and type: exec('error_control_coding.sci', -1);
6. **Run the Simulation:** Execute the script, and Scilab will generate the encoded and decoded messages along with error detection results.
7. **Analyze the Results:** Compare the error detection and correction capabilities of Linear Block and Cyclic Codes under different error conditions.

RESULT: